A Scalability Study and New Algorithms for Large-Scale Many-Objective Optimization

Justin Maltese

Department of Computer Science

Submitted in partial fulfillment
of the requirements of:

Master of Science

Faculty of Mathematics and Science, Brock University
St. Catharines, Ontario

To my parents, whose love and support I will always cherish. Also, to my supervisor Dr. Beatrice Ombuki-Berman, whose invaluable guidance and inspiration made this work possible.

Abstract

Many real-world optimization problems contain multiple (often conflicting) goals to be optimized concurrently, commonly referred to as multi-objective problems (MOPs). Over the past few decades, a plethora of multi-objective algorithms have been proposed, often tested on MOPs possessing two or three objectives. Unfortunately, when tasked with solving MOPs with four or more objectives, referred to as many-objective problems (MaOPs), a large majority of optimizers experience significant performance degradation. The downfall of these optimizers is that simultaneously maintaining a well-spread set of solutions along with appropriate selection pressure to converge becomes difficult as the number of objectives increase. This difficulty is further compounded for large-scale MaOPs, i.e., MaOPs possessing large amounts of decision variables. In this thesis, we explore the challenges of many-objective optimization and propose three new promising algorithms designed to efficiently solve MaOPs. Experimental results demonstrate the proposed optimizers to perform very well, often outperforming state-of-the-art many-objective algorithms.

Acknowledgements

I would like to extend my gratitude to the following people:

- Prof. Beatrice M. Ombuki-Berman for her excellent guidance and supervision throughout the entire duration of this work.
- Prof. Andries P. Engelbrecht for his collaboration during the research process.
- The Computer Science Department at Brock University for their exceptional level of education provided to me.

J.P.M

Contents

1 Introduction **1**

1.1 Objectives and Contributions 4

1.2 Thesis Structure 5

2 Basic Algorithms **7**

2.1 Particle Swarm Optimization 7

2.1.1 Particle Movement 7

2.1.2 Update Equations 8

2.1.3 Weight Values 9

2.2 Differential Evolution 10

2.2.1 Basic Steps 10

2.2.2 Parameter Selection 11

2.3 Genetic Algorithms 13

2.3.1 Fitness Evaluation 13

2.3.2 Mating Selection 14

2.3.3 Crossover 15

2.3.4 Mutation 15

3 Multi-objective Optimization **17**

3.1 Background 17

3.2 Pareto Optimality 19

3.3 Relevant Algorithms 21

3.3.1 NSGA-II 21

3.3.2 SMPSO 24

3.3.3 GDE3 26

3.4 WFG Benchmark Suite 27

4 Many-objective Optimization **30**
4.1 Difficulties . . . 30
4.2 Literature Review . . . 33
4.2.1 Dominance Relation Modification . . . 33
4.2.2 Additional Convergence Metric . . . 37
4.2.3 Performance Indicator Usage . . . 42
4.2.4 Decomposition . . . 43
4.2.5 Non-Pareto Approaches . . . 47

5 Proposed Approaches **52**
5.1 Knee-Driven Algorithms . . . 52
5.1.1 Knee Point Identification . . . 55
5.1.2 KnPSO . . . 56
5.1.3 KnDE . . . 58
5.2 SrEA/D . . . 59

6 Experimental Setup **64**
6.1 Performance Measures . . . 64
6.1.1 Hypervolume Indicator . . . 65
6.1.2 Inverted Generation Distance . . . 66
6.2 Statistical Methods . . . 67
6.2.1 Mann-Whitney-Wilcoxon Rank Sum Test . . . 68
6.3 Benchmark Functions . . . 69
6.4 Algorithm Parameters . . . 69
6.4.1 GA Parameters . . . 71
6.4.2 PSO Parameters . . . 71
6.4.3 DE Parameters . . . 72

7 Knee Influence Experiments **73**
7.1 Hypervolume . . . 73
7.2 Inverted Generational Distance . . . 84

8 Objective Scalability Experiments **96**
8.1 Hypervolume . . . 96
8.2 Inverted Generational Distance . . . 107

9 Decision Variable Scalability Experiments **120**
9.1 Hypervolume 120
9.2 Inverted Generational Distance 130

10 Concluding Remarks and Future Work **140**

Bibliography **154**

Appendices **154**

A Summary of Experiments Performed **155**

List of Tables

6.1 Overview of WFG Functions Used 70
6.2 Number of Reference Points Employed 71
6.3 Number of Function Evaluations Used 71
6.4 T values of KnPSO, KnEA and KnDE 72
6.5 S values of CDAS-SMPSO 72

7.1 Mann-Whitney Wins and Losses For The Hypervolume Metric Using 30 Decision Variables, 2 Objectives 76
7.2 Hypervolume Rank Summary Using 30 Decision Variables, 2 Objectives 76
7.3 Mann-Whitney Wins and Losses For The Hypervolume Metric Using 30 Decision Variables, 4 Objectives 77
7.4 Hypervolume Rank Summary Using 30 Decision Variables, 4 Objectives 77
7.5 Mann-Whitney Wins and Losses For The Hypervolume Metric Using 30 Decision Variables, 6 Objectives 78
7.6 Hypervolume Rank Summary Using 30 Decision Variables, 6 Objectives 78
7.7 Mann-Whitney Wins and Losses For The Hypervolume Metric Using 30 Decision Variables, 8 Objectives 79
7.8 Hypervolume Rank Summary Using 30 Decision Variables, 8 Objectives 79
7.9 Mann-Whitney Wins and Losses For The Hypervolume Metric Using 30 Decision Variables, 10 Objectives 80
7.10 Hypervolume Rank Summary Using 30 Decision Variables, 10 Objectives 80
7.11 Mann-Whitney Wins and Losses For The IGD Metric Using 30 Decision Variables, 2 Objectives 86
7.12 IGD Rank Summary Using 30 Decision Variables, 2 Objectives 86
7.13 Mann-Whitney Wins and Losses For The IGD Metric Using 30 Decision Variables, 4 Objectives 87
7.14 IGD Rank Summary Using 30 Decision Variables, 4 Objectives 87

7.15 Mann-Whitney Wins and Losses For The IGD Metric Using 30 Decision Variables, 6 Objectives 88
7.16 IGD Rank Summary Using 30 Decision Variables, 6 Objectives 88
7.17 Mann-Whitney Wins and Losses For The IGD Metric Using 30 Decision Variables, 8 Objectives 89
7.18 IGD Rank Summary Using 30 Decision Variables, 8 Objectives 89
7.19 Mann-Whitney Wins and Losses For The IGD Metric Using 30 Decision Variables, 10 Objectives 90
7.20 IGD Rank Summary Using 30 Decision Variables, 10 Objectives . . . 90

8.1 Mann-Whitney Wins and Losses For The Hypervolume Metric Using 30 Decision Variables, 2 Objectives 99
8.2 Hypervolume Rank Summary Using 30 Decision Variables, 2 Objectives 99
8.3 Mann-Whitney Wins and Losses For The Hypervolume Metric Using 30 Decision Variables, 4 Objectives 100
8.4 Hypervolume Rank Summary Using 30 Decision Variables, 4 Objectives 100
8.5 Mann-Whitney Wins and Losses For The Hypervolume Metric Using 30 Decision Variables, 6 Objectives 101
8.6 Hypervolume Rank Summary Using 30 Decision Variables, 6 Objectives 101
8.7 Mann-Whitney Wins and Losses For The Hypervolume Metric Using 30 Decision Variables, 8 Objectives 102
8.8 Hypervolume Rank Summary Using 30 Decision Variables, 8 Objectives 102
8.9 Mann-Whitney Wins and Losses For The Hypervolume Metric Using 30 Decision Variables, 10 Objectives 103
8.10 Hypervolume Rank Summary Using 30 Decision Variables, 10 Objectives 103
8.11 Mann-Whitney Wins and Losses For The IGD Metric Using 30 Decision Variables, 2 Objectives 109
8.12 IGD Rank Summary Using 30 Decision Variables, 2 Objectives 109
8.13 Mann-Whitney Wins and Losses For The IGD Metric Using 30 Decision Variables, 4 Objectives 110
8.14 IGD Rank Summary Using 30 Decision Variables, 4 Objectives 110
8.15 Mann-Whitney Wins and Losses For The IGD Metric Using 30 Decision Variables, 6 Objectives 111
8.16 IGD Rank Summary Using 30 Decision Variables, 6 Objectives 111
8.17 Mann-Whitney Wins and Losses For The IGD Metric Using 30 Decision Variables, 8 Objectives 112

8.18 IGD Rank Summary Using 30 Decision Variables, 8 Objectives 112
8.19 Mann-Whitney Wins and Losses For The IGD Metric Using 30 Decision Variables, 10 Objectives 113
8.20 IGD Rank Summary Using 30 Decision Variables, 10 Objectives . . . 113

9.1 Mann-Whitney Wins and Losses For The Hypervolume Metric Using 30 Decision Variables, 10 Objectives 123
9.2 Hypervolume Rank Summary Using 30 Decision Variables, 10 Objectives123
9.3 Mann-Whitney Wins and Losses For The Hypervolume Metric Using 100 Decision Variables, 10 Objectives 124
9.4 Hypervolume Rank Summary Using 100 Decision Variables, 10 Objectives 124
9.5 Mann-Whitney Wins and Losses For The Hypervolume Metric Using 500 Decision Variables, 10 Objectives 125
9.6 Hypervolume Rank Summary Using 500 Decision Variables, 10 Objectives 125
9.7 Mann-Whitney Wins and Losses For The Hypervolume Metric Using 1000 Decision Variables, 10 Objectives 126
9.8 Hypervolume Rank Summary Using 1000 Decision Variables, 10 Objectives 126
9.9 Mann-Whitney Wins and Losses For The IGD Metric Using 30 Decision Variables, 10 Objectives 133
9.10 IGD Rank Summary Using 30 Decision Variables, 10 Objectives . . . 133
9.11 Mann-Whitney Wins and Losses For The IGD Metric Using 100 Decision Variables, 10 Objectives 134
9.12 IGD Rank Summary Using 100 Decision Variables, 10 Objectives . . 134
9.13 Mann-Whitney Wins and Losses For The IGD Metric Using 500 Decision Variables, 10 Objectives 135
9.14 IGD Rank Summary Using 500 Decision Variables, 10 Objectives . . 135
9.15 Mann-Whitney Wins and Losses For The IGD Metric Using 1000 Decision Variables, 10 Objectives 136
9.16 IGD Rank Summary Using 1000 Decision Variables, 10 Objectives . . 136

A.1 Experiments Performed 155
A.1 Experiments Performed (continued) 156
A.1 Experiments Performed (continued) 157
A.1 Experiments Performed (continued) 158

A.1 Experiments Performed (continued) 159
A.1 Experiments Performed (continued) 160
A.1 Experiments Performed (continued) 161
A.1 Experiments Performed (continued) 162
A.1 Experiments Performed (continued) 163
A.1 Experiments Performed (continued) 164
A.1 Experiments Performed (continued) 165
A.1 Experiments Performed (continued) 166
A.1 Experiments Performed (continued) 167
A.1 Experiments Performed (continued) 168
A.1 Experiments Performed (continued) 169
A.1 Experiments Performed (continued) 170
A.1 Experiments Performed (continued) 171
A.1 Experiments Performed (continued) 172
A.1 Experiments Performed (continued) 173
A.1 Experiments Performed (continued) 174
A.1 Experiments Performed (continued) 175
A.1 Experiments Performed (continued) 176
A.1 Experiments Performed (continued) 177
A.1 Experiments Performed (continued) 178
A.1 Experiments Performed (continued) 179
A.1 Experiments Performed (continued) 180
A.1 Experiments Performed (continued) 181
A.1 Experiments Performed (continued) 182
A.1 Experiments Performed (continued) 183
A.1 Experiments Performed (continued) 184
A.1 Experiments Performed (continued) 185
A.1 Experiments Performed (continued) 186
A.1 Experiments Performed (continued) 187
A.1 Experiments Performed (continued) 188
A.1 Experiments Performed (continued) 189
A.1 Experiments Performed (continued) 190
A.1 Experiments Performed (continued) 191
A.1 Experiments Performed (continued) 192
A.1 Experiments Performed (continued) 193

List of Figures

3.1 A Pareto front for a maximization problem with two objectives. . . . 20

4.1 Calculation of the F_{sum} and F_{min} aggregation methods are demonstrated for a front containing five solutions. The MOP in question possesses three objectives. 49

5.1 Eight solutions are shown for a bi-objective MOP. Each non-dominated solution is displayed with a gray fill. Here, solution C is a knee point due to its large marginal rates of return in comparison to the other non-dominated solutions. 53

5.2 The spread-preservation advantages of the subregion concept is exemplified for a population of ten solutions. The four best solutions according to only the NF_{sum} relation are shown on the left, whereas the four most desirable solutions according to SrEA/D, which employs NF_{sum} in each individual subregion, is shown on the right. The selected best solutions are displayed with a gray fill. 60

7.1 Ranking of hypervolume metric vs. number of objectives for the WFG1 problem 81

7.2 Ranking of hypervolume metric vs. number of objectives for the WFG2 problem 81

7.3 Ranking of hypervolume metric vs. number of objectives for the WFG3 problem 81

7.4 Ranking of hypervolume metric vs. number of objectives for the WFG4 problem 82

7.5 Ranking of hypervolume metric vs. number of objectives for the WFG5 problem 82

7.6 Ranking of hypervolume metric vs. number of objectives for the WFG6 problem 82

7.7 Ranking of hypervolume metric vs. number of objectives for the WFG7 problem 83
7.8 Ranking of hypervolume metric vs. number of objectives for the WFG8 problem 83
7.9 Ranking of hypervolume metric vs. number of objectives for the WFG9 problem 83
7.10 Ranking of IGD metric vs. number of objectives for the WFG1 problem 91
7.11 Ranking of IGD metric vs. number of objectives for the WFG2 problem 91
7.12 Ranking of IGD metric vs. number of objectives for the WFG3 problem 91
7.13 Ranking of IGD metric vs. number of objectives for the WFG4 problem 92
7.14 Ranking of IGD metric vs. number of objectives for the WFG5 problem 92
7.15 Ranking of IGD metric vs. number of objectives for the WFG6 problem 92
7.16 Ranking of IGD metric vs. number of objectives for the WFG7 problem 93
7.17 Ranking of IGD metric vs. number of objectives for the WFG8 problem 93
7.18 Ranking of IGD metric vs. number of objectives for the WFG9 problem 93
7.19 Sample Pareto front produced by GDE3 on WFG1. 94
7.20 Sample Pareto front produced by SMPSO on WFG1. 94
7.21 Sample Pareto front produced by NSGA-II on WFG1. 94
7.22 Sample Pareto front produced by KnDE on WFG1. 95
7.23 Sample Pareto front produced by KnPSO on WFG1. 95
7.24 Sample Pareto front produced by KnEA on WFG1. 95

8.1 Ranking of hypervolume metric vs. number of objectives for the WFG1 problem 104
8.2 Ranking of hypervolume metric vs. number of objectives for the WFG2 problem 104
8.3 Ranking of hypervolume metric vs. number of objectives for the WFG3 problem 104
8.4 Ranking of hypervolume metric vs. number of objectives for the WFG4 problem 105
8.5 Ranking of hypervolume metric vs. number of objectives for the WFG5 problem 105
8.6 Ranking of hypervolume metric vs. number of objectives for the WFG6 problem 105
8.7 Ranking of hypervolume metric vs. number of objectives for the WFG7 problem 106

8.8 Ranking of hypervolume metric vs. number of objectives for the WFG8 problem 106
8.9 Ranking of hypervolume metric vs. number of objectives for the WFG9 problem 106
8.10 Ranking of IGD metric vs. number of objectives for the WFG1 problem 114
8.11 Ranking of IGD metric vs. number of objectives for the WFG2 problem 114
8.12 Ranking of IGD metric vs. number of objectives for the WFG3 problem 114
8.13 Ranking of IGD metric vs. number of objectives for the WFG4 problem 115
8.14 Ranking of IGD metric vs. number of objectives for the WFG5 problem 115
8.15 Ranking of IGD metric vs. number of objectives for the WFG6 problem 115
8.16 Ranking of IGD metric vs. number of objectives for the WFG7 problem 116
8.17 Ranking of IGD metric vs. number of objectives for the WFG8 problem 116
8.18 Ranking of IGD metric vs. number of objectives for the WFG9 problem 116
8.19 Sample Pareto front produced by CDAS-SMPSO on WFG1. 117
8.20 Sample Pareto front produced by NSGA-III on WFG1. 117
8.21 Sample Pareto front produced by MOEA/D on WFG1. 117
8.22 Sample Pareto front produced by SrEA/D on WFG1. 118
8.23 Sample Pareto front produced by KnDE on WFG1. 118
8.24 Sample Pareto front produced by KnPSO on WFG1. 118
8.25 Sample Pareto front produced by KnEA on WFG1. 119

9.1 Ranking of hypervolume metric vs. number of decision variables for the WFG1 problem 127
9.2 Ranking of hypervolume metric vs. number of decision variables for the WFG2 problem 127
9.3 Ranking of hypervolume metric vs. number of decision variables for the WFG3 problem 127
9.4 Ranking of hypervolume metric vs. number of decision variables for the WFG4 problem 128
9.5 Ranking of hypervolume metric vs. number of decision variables for the WFG5 problem 128
9.6 Ranking of hypervolume metric vs. number of decision variables for the WFG6 problem 128
9.7 Ranking of hypervolume metric vs. number of decision variables for the WFG7 problem 129

9.8 Ranking of hypervolume metric vs. number of decision variables for the WFG8 problem 129
9.9 Ranking of hypervolume metric vs. number of decision variables for the WFG9 problem 129
9.10 Ranking of IGD metric vs. number of decision variables for the WFG1 problem 137
9.11 Ranking of IGD metric vs. number of decision variables for the WFG2 problem 137
9.12 Ranking of IGD metric vs. number of decision variables for the WFG3 problem 137
9.13 Ranking of IGD metric vs. number of decision variables for the WFG4 problem 138
9.14 Ranking of IGD metric vs. number of decision variables for the WFG5 problem 138
9.15 Ranking of IGD metric vs. number of decision variables for the WFG6 problem 138
9.16 Ranking of IGD metric vs. number of decision variables for the WFG7 problem 139
9.17 Ranking of IGD metric vs. number of decision variables for the WFG8 problem 139
9.18 Ranking of IGD metric vs. number of decision variables for the WFG9 problem 139

List of Algorithms

1 Standard GBest PSO 9
2 DE/rand/1/bin Trial Position Creation 11
3 Pareto Ranking 21
4 NSGA-II 24
5 SOD-CNT 39
6 NSGA-III Niching Procedure 47
7 KnPSO Archive Maintenance 57
8 KnDE Population Reduction 59
9 SrEA/D Basic Framework 61
10 SrEA/D Mating Selection 62
11 SrEA/D Population Truncation 63

Chapter 1

Introduction

Since the beginning of time, humanity has been forced to solve challenging problems in order to ensure survival or gain an advantage over others. Whether it is finding the ideal hiding spot far from predators or choosing the best set of stocks to invest in, problems which demand solutions have arisen naturally and spontaneously in everyday human life throughout the ages. Historically, humankind has had no choice but to solve problems themselves, inherently limiting solutions to the capabilities of the human brain. However, the advent of the personal computer in 1975 has allowed humans to delegate the solving of non-trivial problems to automated algorithmic procedures, completely changing the feasibility of many problems. Problems that were previously impossible to solve by human intellect alone could now be tackled given enough computational power.

Optimization problems, defined as problems where the best solution from the set of all feasible solutions is to be found, is one such category that has benefitted immensely from the inception of computers. Many optimization problems which are easily solved by computational methods would take months or even years of dedicated human effort to determine optimal solutions. However, mankind is still very far from claiming that all optimization problems can be solved efficiently using computers, since the most difficult instances possess unique challenges which are

non-trivial for optimizers to combat. One such property which often increases the difficulty of optimization problems considerably is the presence of multiple distinct objectives requiring concurrent optimization. Problems exhibiting this property are referred to as multi-objective problems (MOPs), commonly encountered in fields such as engineering [109], business [15], mathematics [56] and physics [30]. It is common for objectives within these problems to conflict, making it difficult (and often impossible) for all objectives to be fully optimized simultaneously. To illustrate this phenomenon, suppose that a fellow named Bob desires to purchase a vehicle from a car dealership. Bob prefers a fast vehicle since he enjoys racing with friends, but Bob also desires a vehicle which possesses a good safety rating. Here, Bob is attempting to maximize both top speed and overall car safety, which are two logically conflicting objectives. Bob will likely need to make some sort of trade-off when purchasing the vehicle.

The abundance of practical MOPs presents a real need for effective multi-objective optimizers (MOOs), thus unsurprisingly a large amount of effort has been devoted to proposing novel computational intelligence (CI) algorithms which solve MOPs. A small sample of evolutionary multi-objective optimization (EMO) can be viewed in [28, 106, 114, 62, 7]. Several examples of swarm intelligence MOOs can be seen in [78, 82, 95, 67, 16]. A large majority of previously proposed CIMOOs evaluate their performance on MOPs possessing two or three objectives, often producing a set of well-converged, well-spread solutions near or even on the Pareto-optimal front. Unfortunately, nearly all of the classic CIMOOs have been shown to degrade when the number of objectives are increased, especially those which perform optimization using the Pareto-dominance relation [52, 92]. Problems possessing more than three objectives, commonly referred to as many-objective problems (MaOPs), present a serious challenge for MOOs. Possessing the ability to efficiently solve MaOPs is highly desirable for CIMOOs, since many real-world applications such as industrial scheduling [99, 105], automotive engine calibration problems [71] and hybrid car controller

optimization [77] have more than three objectives.

The poor scalability of CIMOOs is due to a variety of challenges unique to MaOPs. The most significant issue experienced when attempting to solve MaOPs is the difficulty in balancing convergence and diversity. Since the end goal of any MOO is to obtain an approximation of the true Pareto-optimal front, selection pressure to converge along with a focus on solution spread must be concurrently maintained throughout optimization. Unfortunately, prioritizing both convergence and diversity becomes increasingly more difficult as the number of objectives grow. Consequently, most CIMOOs end up sacrificing convergence for solution spread, or vice-versa.

The end result of optimization is thus either a well-spread set of solutions which are undesirably far from Pareto-optimal, or a solution set which has converged to a small subregion near or on the Pareto-optimal front. In the case of traditional Pareto-based optimizers, the former is often the case. The main reason for this phenomenon is a loss of selection pressure towards the true Pareto-optimal front. As the number of objectives grow, the Pareto-dominance relation essentially loses the ability to distinguish desirable solutions, since nearly all population members are non-dominated at an early stage of the search [53]. In fact, over 90% of a randomly generated initial solution set is non-dominated when the number of objectives are 8 or more [44]. Therefore, using only the Pareto-dominance relation as selection criteria would nearly be a completely random search, likely guiding a CIMOO into sub-optimal areas of the search space.

Many other challenges exist for MaOPs such as the exponentially increasing search space, difficulty in maintaining a uniformly spread solution set, inability to visualize the trade-off surface, difficulty when selecting a final solution and increasing complexity of performance indicators. While each of these are important, the difficulty in balancing convergence and diversity is the main motivating factor for this work. Thus, our goal is to propose new many-objective algorithms which will efficiently

converge to the true Pareto-optimal front while simultaneously maintaining a uniformly spread set of solutions. The first two algorithms proposed in this work are based on the concept of knee points, recently shown to be useful for MaOPs [108]. Knee points, defined as Pareto front solutions for which an improvement in one objective will severely reduce the desirability of one or more objectives, are incorporated into a particle swarm optimization (PSO) and differential evolution (DE) approach. Since incorporating knee points during optimization can be seen as a bias towards a higher hypervolume value, these algorithms should maintain selection pressure to converge towards the true Pareto front, even when tasked with solving MaOPs. The third many-objective optimizer proposed in this work is a hybrid evolutionary algorithm which abandons the Pareto dominance relation entirely, using a sum-of-ranks approach [6] to promote convergence along with promising decomposition concepts [69] to maintain solution spread. We compare our proposed algorithms with other state-of-the-art many objective optimizers over nine challenging functions to obtain an idea of the overall effectiveness of each method.

1.1 Objectives and Contributions

This thesis proposes to achieve a wide variety of goals. These goals, along with their respective contributions, can be formulated as follows:

- Explore the challenges of many-objective optimization, providing further insight into the unique difficulties of MaOPs.

- Establish the benefit, if any, that using knee points brings in helping solve MaOPs. For this purpose, knee points are added to several existing MOOs and performance is evaluated both with and without knee point incorporation. One should note that knee points were applied to an evolutionary algorithm

in [108], however comparisons determining the exact performance gain of knee points were not performed.

- Determine the competitiveness of our proposed hybrid evolutionary strategy when compared against other state-of-the-art many objective optimizers.

- Perform a scalability study which helps give insight into the current state of many-objective optimization. For this purpose, we employ our proposed approaches along with other state-of-the-art many objective optimizers and compare performance over an increasing number of objectives to evaluate scalability. To the best of the authors knowledge, there is currently no literature which compares PSO, DE and genetic algorithm (GA) many-objective approaches. Thus, this thesis aims to fill the current literature gap.

- Analyze the performance of each many-objective algorithm for MaOPs possessing a large number of decision variables, referred to as large-scale MaOPs. Since practical MaOPs may contain a large number of decision variable, it is desirable for many-objective optimizers to also scale well when tasked with solving large-scale MaOPs. One should note that to the best of the authors knowledge, no literature currently exists on evaluating CIMOOs using large-scale MaOPs.

- Determine the weaknesses and strengths of each tested CIMOO with respect to function shapes and modalities. This will be accomplished through providing thorough analysis of each optimizer over a diverse set of challenging problems.

1.2 Thesis Structure

The remainder of this thesis is organized as follows:

Chapter 2 contains background information on basic single-objective variants of PSO, DE and GA. This includes detailed implementation of each algorithm.

Chapter 3 provides an overview of multi-objective optimization along with several algorithms designed to tackle MOPs. The concept of Pareto dominance is also discussed.

Chapter 4 discusses concepts related to many-objective optimization. A review of the current literature is presented in this chapter. Additionally, knee points are introduced and explained.

Chapter 5 gives implementation-level details on each algorithm proposed within this work.

Chapter 6 describes the experimental setup used in this study. An overview of the benchmark functions used in the study is provided. Multiple performance measures utilized within this work are also detailed. Parameter sets for all algorithms are given. Finally, the statistical methods of analysis are described.

Chapter 7 investigates the exact performance gain of using knee points. Knee-driven algorithms are compared against their non-knee counterparts over a variety of functions.

Chapter 8 presents results of the experiments analyzing algorithm performance as the number of objectives increase. Analysis and discussion of the results is also performed.

Chapter 9 contains an overview of experiments which evaluate the scalability of each many-objective optimizer with respect to an increasing number of decision variables.

Chapter 10 discusses conclusions drawn and suggests avenues for future work.

Chapter 2

Basic Algorithms

This chapter provides an overview of several basic optimization algorithms relevant to this work. Algorithms covered include PSO, DE and GAs.

2.1 Particle Swarm Optimization

PSO, introduced by Kennedy and Eberhart in [55], is a stochastic metaheuristic optimization algorithm. Modelled after behaviours observed in bird flocks, the PSO algorithm serves as a powerful optimizer inherently designed for problems possessing a continuous domain. PSO has been successfully applied to many practical applications including the optimization of security identification systems [54], electromagnetic devices [46], antennas [70] and power plants [45].

2.1.1 Particle Movement

Within the PSO algorithm, a population of simple entities known as *particles* is maintained. The position of each particle represents a candidate solution to a given problem optimized collectively by the swarm. Particle position desirability is determined by a quality evaluation function relating to the problem at hand. Particles navigate the search space by iteratively flocking around desirable positions with the

intent of uncovering the globally optimal position(s) in the search space. Movement of a particle p is primarily influenced by three factors:

- The current velocity of p
- The personal best position of p
- The neighbourhood best position of p

Weight values are assigned to the above factors, affecting the level of influence each has. The *inertial* weight is responsible for controlling particle velocity influence, the *cognitive* weight controls influence of the personal best position and the *social* weight varies the influence level of the neighbourhood best position.

2.1.2 Update Equations

Pseudocode for the standard global best (GBest) PSO algorithm is shown in Algorithm 1. Particle positions are updated in a synchronous fashion [13]. One should note that this pseudocode uses a star neighbourhood topology [55], employing the social component to draw particles towards the overall best position of the entire swarm. It is also possible to create neighbourhoods of particle attraction via the use of a ring topology [32]. Algorithm 1 makes use of two update equations on lines 12 and 13 defined as:

$$S.\vec{v}_i(t+1) = \omega S.\vec{v}_i(t) + c_1\vec{r}_1(S.\vec{y}_i(t) - S.\vec{x}_i(t)) + c_2\vec{r}_2(S.\vec{\hat{y}}(t) - S.\vec{x}_i(t)) \tag{2.1}$$

$$S.\vec{x}_i(t+1) = S.\vec{x}_i(t) + S.\vec{v}_i(t+1) \tag{2.2}$$

where $S.\vec{x}$ corresponds to the current position of particle i in swarm S, $S.\vec{v}_i$ refers to the velocity of particle i in swarm S, $S.\vec{y}_i$ is the personal best position of particle i in swarm S, $S.\vec{\hat{y}}$ is the global best position of swarm S, ω is the inertia weight, c_1 is the

Algorithm 1 Standard GBest PSO

Create and initialize a swarm, S, with candidate solutions in n_x dimensions
while *termination criterion not satisfied* **do**
    for each particle i in S **do**
        if $f(\mathrm{S}.\vec{x}_i) < f(\mathrm{S}.\vec{y}_i)$ **then**
            $\mathrm{S}.\vec{y}_i = \mathrm{S}.\vec{x}_i$
        end if
        if $f(\mathrm{S}.\vec{y}_i) < f(S.\vec{\hat{y}})$ **then**
            $\mathrm{S}.\vec{\hat{y}} = \mathrm{S}.\vec{y}_i$
        end if
    end for
    for each particle i in S **do**
        Update velocity of particle i using Equation (2.1)
        Update position of particle i using Equation (2.2)
    end for
end while

cognitive weight, c_2 represents the social weight, and $\vec{r}_1$ and $\vec{r}_2$ are vectors consisting of uniformly distributed random numbers within the range [0,1]. Note that these update equations are meant for continuous values, thus they can be inefficient when applied directly to discrete-valued problems. Several modifications of the update equations have been introduced in previous literature to accommodate discrete domains, e.g., see [48, 93, 87].

2.1.3 Weight Values

The weights in Equation 2.4 are commonly set to values conforming to Clerc and Kennedy's equations in [17] which are:

$$S.\vec{v}_i(t+1) = K(S.\vec{v}_i(t) + c_1\vec{r}_1(S.\vec{y}_i(t) - S.\vec{x}_i(t)) + c_2\vec{r}_2(S.\vec{\hat{y}}(t) - S.\vec{x}_i(t))) \quad (2.3)$$

Note that K is a constriction factor calculated as

$$K = \frac{2}{|2 - \phi - \sqrt{\phi^2 - 4\phi}|} \tag{2.4}$$

where ϕ is calculated as $(c_1 + c_2)$ and must be greater than 4. By distributing K in Equation 2.6, the missing inertial weight can be included. Choosing $\phi = 4.1$ to satisfy Clerc's formula has been shown to perform well [14], which corresponds to the values $\alpha = 0.729$, $\omega = 1.494$ and $\lambda = 1.494$. These parameter values are commonly seen in PSO literature, however other values have also been proposed [94, 100].

2.2 Differential Evolution

DE, introduced in [98], is a simple yet efficient evolutionary computation technique which optimizes a problem by iteratively improving a set of candidate solutions. Within DE, candidate solutions are referred to as *agents* and the set of maintained agents is known as a *population*. Agents within the population, initially placed at random positions in the search space, are updated by combining a number of agents from the population. Like a typical evolutionary algorithm, DE performs optimization using selection, crossover and mutation operators. Concerning stopping criterion, DE continues optimization until a maximum number of generations are reached or an ideal solution is encountered.

2.2.1 Basic Steps

At each generation, a basic DE algorithm performs the following steps for an agent $\vec{x}$ in the population P:

1. Select an agent $\vec{a}$ from P as the *target vector*

2. Select one or more agents from P to serve as *difference vectors*

Algorithm 2 DE/rand/1/bin Trial Position Creation

1: $\vec{a} \in P, \vec{a} \neq \vec{x}$
2: $\vec{b} \in P, \vec{b} \neq \vec{x} \wedge \vec{b} \neq \vec{a}$
3: $\vec{c} \in P, \vec{c} \neq \vec{x} \wedge \vec{c} \neq \vec{b} \wedge \vec{c} \neq \vec{a}$
4: $i_{rand} \in \{1, 2, ..., D\}$
5: **for** $(i = 1; i \leq D; i = i + 1)$ **do**
6: **if** $rand() < CR \vee i = i_{rand}$ **then**
7: $\vec{t_i} = \vec{a_i} + F \cdot (\vec{b_i} - \vec{c_i})$
8: **else**
9: $\vec{t_i} = \vec{x_i}$
10: **end if**
11: **end for**

3. Mutate the target vector in some fashion using the difference vectors. The resulting vector $\vec{t}$ is deemed the *trial position*

4. Apply a crossover operator to combine $\vec{x}$ and $\vec{t}$. The resulting offspring replaces $\vec{x}$ if it has better fitness than $\vec{x}$

Note that DE is considered an elitist strategy, since the average fitness value of the population will never decrease between generations. While a variety of DE variants exist, the most commonly used version is DE/rand/1/bin [84]. Creation of a trial position $\vec{t}$ for an agent $\vec{x}$ in DE/rand/1/bin is shown in Algorithm 2. Note that crossover and mutation are performed together in DE/rand/1/bin (see line 7 in Algorithm 2). Within Algorithm 2 P represents the current population, D is the dimensionality of the problem to be optimized, $rand()$ is a function which returns a random floating point value in the range $[0, 1)$, CR is the *crossover probability* and F is a scaling factor for mutation. Both CR and F are user-defined parameters, further discussed in Section 2.3.2.

2.2.2 Parameter Selection

Three main parameters exist in DE, defined *a priori* to optimization. The population size, denoted NP, refers to the number of agents that will be in the population at each

generation. The crossover probability, CR, is a value between [0,1] which controls the crossover operation, essentially representing the chance that a dimension of the trial position will be chosen from a linear combination of three randomly chosen agents. However, at least one randomly chosen dimension of the trial position is guaranteed to be a crossover result, seen from the condition $i = i_{rand}$ on Line 6 in Algorithm 2. In practice, CR controls the level of exploration seen in the search.

Another crucial parameter to DE is the differential weight, F. F acts as a scaling factor for mutation, subject to $F > 0$. Mutation within DE is self-adaptive to the problem surface similar to Covariance Matrix Adaptation Evolutionary Strategies [42]. In early generations, mutation magnitude is large due to agents typically having radically different positions in the search space. As evolution continues, the population converges and agents become more similar, reducing the overall mutation magnitude. Choosing an appropriate value of F is critical, as it controls the speed and robustness of the search. The problem of premature convergence into local minima is overcome by choosing a suitably large value of F. However, choosing values that are too large may lead to erratic behaviour and a severely decreased convergence rate.

Since the choice of control parameters are critical to performance of the DE algorithm, a variety of literature exists on the topic. Ali and Törn [1] propose an adaptive strategy for the differential weight F, which changes during evolution according to the equation

$$F = \begin{cases} max(F_{min}, 1 - |\frac{f_{max}}{f_{min}}|), & \text{if } |\frac{f_{max}}{f_{min}}| < 1. \\ max(F_{min}, 1 - |\frac{f_{min}}{f_{max}}|), & \text{otherwise} \end{cases} \tag{2.5}$$

where f_{min} is the minimum fitness value in the population, f_{max} is the maximum fitness value in the population and F_{min} is an input parameter such that $F \in [F_{min}, 1]$. This strategy is used to create a more diversified search early on and an intensified search as the population is converging in later generations. Previous literature [1]

has shown this method to be quite effective despite its simplicity. However, one should note that when $f_{min} < 0$ and $f_{max} > 0$, F experiences large fluctuations and the method loses its effectiveness. Several other methods have been proposed for adaptive parameter control in DE, such as [9, 81, 85].

2.3 Genetic Algorithms

GAs, proposed by James Holland in [47], are a family of population-based optimization techniques inspired by biological evolution. Like other metaheuristic techniques, they are not guaranteed to produce optimal solutions, rather they are often employed to obtain good approximations on challenging problems in a reasonable amount of time. Survival of the fittest is a central idea in GAs, as fitter individuals are more likely to mate and persist into future generations, mimicking the process of natural selection in nature. GAs have been applied to a wide variety of problems, e.g., vehicle routing [80], bin-packing [86], job scheduling [38] and control engineering [68].

Within GAs, a *population* of candidate solutions is maintained, where each individual candidate solution is referred to as a *chromosome*. Chromosomes possess a fixed number of dimensions, each referred to as an individual *gene*. Arguably the most challenging aspect of GAs is choosing how to represent candidate solutions as chromosomes, a process referred to as *encoding*. Determining a suitable encoding is problem specific, with certain problems possessing relatively more obvious encodings than others. The simplest forms of encoding are seen for problems which can be represented as binary strings [40].

2.3.1 Fitness Evaluation

To determine the quality of an individual chromosome, it is necessary to use an evaluation function which produces a single scalar value. The resultant value is referred

to as the *fitness* of the individual and represents their desirability with respect to the problem at hand. Thus, fitness functions are entirely problem dependent. In the case of a minimization problem lower fitness values are deemed better, whereas higher fitness values are more desirable for maximization problems.

2.3.2 Mating Selection

At each generation in a GA, individuals are mated together to produce offspring which persist into the next generation. Selection of parents must be done in such a way that promotes fit individuals while also still allowing less fit members a chance to be chosen for diversity maintenance purposes. If the current best individuals are selected to produce every single child in the next generation, the GA may become susceptible to premature convergence into local minima.

In general, selection schemes either rate the fitness of all individuals and preferentially select from them, or retrieve a random sample of individuals and choose the best [41]. The most common selection strategies are:

- **Tournament Selection** [10] - k individuals are randomly selected out of the current population to participate in the tournament. From these individuals, the winner of the tournament is deemed to be the individual with the best fitness function value. The value of n determines the amount of selection pressure, with larger tournaments essentially leading to more selection pressure.

- **Roulette Wheel Selection** [23] - Each individual in the current population is assigned space on a roulette wheel proportionate to their fitness. More fit individuals occupy a larger portion of the wheel, giving them a higher chance of being selected. This type of selection is also referred to as *fitness proportionate selection*.

2.3.3 Crossover

To perform the mating operation, two individuals are essentially combined by exchanging genes to form children. This recombination process is formally referred to as *crossover*. Crossover essentially serves as the main recombination operator within GAs, used to produce an offspring population at each generation. While many different crossover operators have been proposed in previous literature, choice of crossover type is often dependent on the problem at hand as permutation problems may require specialized crossovers. Examples of some common crossovers are the n-point [5], uniform-order [76], cycle [79] and partially-mapped [79] crossovers.

To give the reader a better idea of the crossover process, we exemplify the n-point crossover with $n = 1$. This operator is performed by randomly choosing a point in a chromosome string and exchanging all genes after that point using two selected parents. As an example, let $p_1 = 10011100$ and $p_2 = 11101011$ be two parents selected for crossover. Assume that the crossover point is randomly chosen to be after the fourth gene. Then, the two children produced would be

$$c_1 = 1001|1011$$

$$c_2 = 1110|1100$$

2.3.4 Mutation

To overcome local minima and improve the searching capabilities of a GA, a *mutation* operator is used to perform a random minor alteration of a chromosome. Unlike the crossover operator, mutation requires only one selected individual. Mutation is used to increase exploitation, providing additional diversity and aiding in the prevention of premature convergence. An example of mutation would be the bit flip method [76], which simply selects a random position within a binary chromosome and flips the bit

at the selected position. Several other types of mutation operators exist such as the inversion [75] and exchange [75] methods.

Chapter 3

Multi-objective Optimization

This chapter provides an overview of multi-objective optimization. Topics covered include pertinent background information, Pareto optimality, several relevant multi-objective optimization algorithms and a multi-objective benchmarking suite employed in this work.

3.1 Background

Multi-objective optimization refers to the simultaneous optimization of two or more objectives. A common difficulty seen in MOPs is conflicting objectives, where trade-offs must be made in some objective(s) in order to further optimize other objectives. Consequently, for many MOPs it becomes extremely difficult or even impossible to obtain a utopian solution which fully optimizes all objectives concurrently. Multi-objective optimization problems[1] can be formally expressed as

$$\text{minimize } \mathbf{F}(\vec{x})$$

$$\text{subject to } \vec{x} \in [x_{min}, x_{max}]^{n_x}$$

[1]Minimization problems are assumed for this section. When dealing with maximization problems, all objectives would be maximized as opposed to minimized.

where $\mathbf{F}(\vec{x}) = f_1(\vec{x}), f_2(\vec{x}), ..., f_{n_c}(\vec{x}))$, $\vec{x} = (x_1, x_2, ..., x_{n_x})$; n_x refers to the dimensionality of the search space and n_c represents the number of objectives. Since many optimization algorithms are designed for problems with only one objective, they cannot be applied for multi-objective optimization without modification. The most straightforward method of modifying algorithms to solve MOPs is to simply sum the objective functions together with no regard for how important each objective is, transforming a MOP into a single-objective problem. This technique is often improved via assigning a multiplicative weighting to each objective function before summation, ideally using the preferences of a decision maker. This essentially defines the overall fitness of a solution as a summation of all objectives, with certain objectives having higher priority than others during the search. An example of this approach can be found in [83]. Within weighted summation approaches, the weighting assigned to each objective can either be static or dynamically updated during optimization. Both approaches have been shown to be successful [83].

As a consequence of the conflicting nature of objectives commonly seen in MOPs, a decision maker is often required to accept trade-offs by prioritizing certain objectives over others. Several approaches to prioritizing objectives are possible. The *a priori* approach deals with assigning objective bias before optimization occurs, allowing the algorithm to prioritize a regions of the search space. Domain knowledge can also be incorporated when assigning objective biases. This approach generally produces a small yet highly specialized set of solutions for the decision maker.

The opposite of an *a priori* approach would be an *a posteriori* approach in which trade-offs would be made manually by the decision maker after the selected optimization algorithm has already produced a set of solutions. *A posteriori* approaches allow a decision maker to select from a large set of non-specialized solutions, ideally possessing the best possible set of trade-offs. A significant drawback of this approach is that challenging problems may present significant difficulty for a MOO in obtaining

a complete approximation of the best solutions of the entire search space. A balanced approach which yields solution sets that are not highly specialized or generalized is known as *interactive*. Here, the decision maker would interact with a MOO during the optimization process, dynamically guiding the optimizer into desired areas of the search space.

3.2 Pareto Optimality

Pareto optimality in its most general form refers to a state of resource distribution in which a single individual's resources cannot improve without degrading at least one other individual's resources. Originally referred to as Pareto efficiency, the concept of Pareto optimality originated from economic observations by Vilfredo Pareto (1848 - 1923). Note that the concept of Pareto optimality continues to be applied within the field of economics, e.g., see [97].

Within the context of multi-objective optimization, the term Pareto-optimal is used to refer to solutions which cannot improve any objective further without worsening any other objective. Obtaining the set of Pareto-optimal solutions, referred to as the Pareto front, is the end goal of a multi-objective optimization algorithm. Solutions which do not belong to the Pareto front are undesirable due to the fact that one or more objectives can be further optimized without degrading other objectives. The Pareto front is commonly analyzed by a decision maker with domain knowledge, as solutions belonging to the Pareto front contain trade-offs in one or more objectives.

A formal definition [110] of the Pareto Front is as follows:

$$PF^* = \{\vec{y^*} \in A | \nexists \vec{y} \in A : \vec{y} \prec \vec{y^*}\} \tag{3.1}$$

where $\prec$ is a strict dominance relation such that $\vec{y} \prec \vec{y^*}$ if $\vec{y_i} \leq \vec{y_i^*}$ for all i and $\vec{y_i} < \vec{y_i^*}$ for some i. A denotes the entire range of all dependent variables, referred to as the

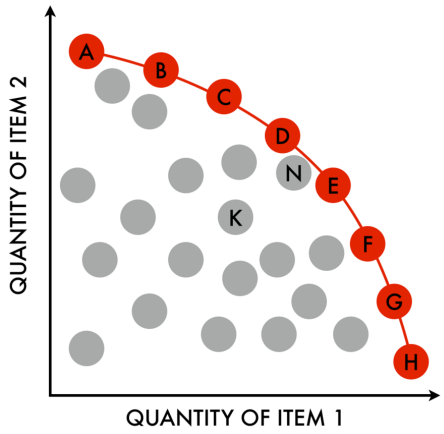

Figure 3.1: A Pareto front for a maximization problem with two objectives.

objective space. A sample Pareto front is shown in Figure 2.1[2]. The front itself is highlighted in red and represents all solutions that are not dominated by any other solutions. Note that solution K is not part of the Pareto front as it is dominated by both solution D and E.

Using the Pareto dominance relation, solutions can be assigned a *Pareto rank* value. Algorithm 3 exemplifies how Pareto ranks can be computed for a solution set P. First, the set of all non-dominated solutions in P, denoted F, are given a rank of 1. Next, all solutions in F are removed from P and F is reinitialized. Thereafter, any remaining non-dominated solutions in P are given a rank of 2 and placed into F. This process is repeated until the entire solution set is ranked. Note that Pareto ranking does not produce a single "best" solution, unless only one non-dominated solution is present in the entire population. Rather, Pareto ranking produces a set of rank 1 solutions, where no rank 1 solution can be said to be strictly better than another rank 1 solution using only the Pareto dominance relation.

[2]Original author is Noel Rosario. Image is used under the Creative Commons Attribution-Share Alike 3.0 Unported License.

Algorithm 3 Pareto Ranking

```
1: current_rank = 1
2: while |P| ≠ 0 do
3:     F = ∅
4:     for i = 1 to |P| do
5:         if non_dominated(P_i) then
6:             F = F ∪ P_i
7:             P_i.rank = current_rank
8:         end if
9:     end for
10:    P = P − F
11:    current_rank = current_rank + 1
12: end while
```

3.3 Relevant Algorithms

While there exists a multitude of algorithms designed for multi-objective optimization, a considerable portion of MOOs proposed within the last few decades use the Pareto dominance relation. Non-dominated solutions are archived upon their uncovering and the search is guided around a subset of non-dominated vectors. This section contains information on several popular Pareto-based PSO, GA and DE algorithms which are relevant to this work.

3.3.1 NSGA-II

The non-dominated sorting genetic algorithm (NSGA) [96] was one of the first evolutionary algorithms to perform optimization based on the Pareto dominance relation. Prior to NSGAs introduction, most multi-objective algorithms performed optimization by converting MOPS to single objective problems and emphasizing one particular Pareto-optimal solution at a time. The NSGA algorithm sought to improve this by searching for the entire set of Pareto-optimal solutions, since each is regarded as equally important in the absence of problem specific information.

The NSGA algorithm uses a method referred to as non-dominated sorting to separate the current Pareto front into K individual fronts based on the dominance relation.

This process is performed as follows: Within the current Pareto front P_f, the set of non-dominated solutions is retrieved and forms the initial front F_0. These solutions are then removed from P_f and the set of non-dominated solutions is recalculated, forming the next front F_1. This process repeats until there are no solutions left in P_f, returning a set of fronts F_0, F_1,..,F_K.

The overall complexity of the naive non-dominated sorting procedure is $O(MN^3)$, where M corresponds to the number of objectives to be optimized and N is the population size. Determining the set of non-dominated solutions within each front consists of comparing each solution with all other solutions over all objectives, giving $O(MN^2)$ total comparisons. In the worst case there are N fronts with a single solution in each, thus in total the complexity of the entire procedure would be $O(MN^3)$. This essentially makes NSGA computationally inefficient, since the costly non-dominated sorting operation must be repeated at each generation.

The NSGA-II algorithm introduced in [28] was proposed as an improvement to the original NSGA algorithm. NSGA-II improves the computational efficiency of NSGA by using a better non-dominated sorting procedure. This method maintains a list s_p for each solution s, where s_p contains all of the solutions dominated by s. Additionally, each solution maintains a domination count corresponding to the number of solutions that they are dominated by. The first front is formed using all solutions which have an initial domination count of zero. Next, all solutions contained within each s_p of the first front have their domination count decreased by one. If at any point the dominance count of a solution becomes zero the corresponding solution is added to a new list W representing the second front. The above procedure is then repeated for all members of W, forming the third front. This continues until all solutions have been placed in a front. The computational complexity of this method is improved to $O(MN^2)$, however the storage requirement increases to $O(N^2)$.

The NSGA-II algorithm introduces the use of the crowding distance metric, which

acts as a parameterless diversity maintenance operator. The crowding distance of a solution is calculated by forming a hypercube using its nearest neighbours as vertices. For each objective, the closest solutions on either side of the target solution are regarded as the nearest neighbours. This requires sorting solutions by each objective. The crowding distance is then computed as the average side length of the created hypercuboid. One should note that calculation of the crowding distance metric should only be done using solutions from the same non-dominated front.

Another improvement of NSGA-II is the use of elitism, which has been demonstrated to speed up performance of GAs significantly [111]. Elitism is implemented by combining the current population of size N with the previous best members, forming a population of size $2N$. These solutions are then compared and the best N solutions are kept as parents which will be used to generate a new population. This procedure ensures that the most desirable solutions will continue to proceed through evolution. The entire NSGA-II algorithm is shown in Algorithm 4. Here P_t represents the set of parents at time t, Q_t is the population at time t, R_t is the combination of P_t and Q_t whose best members are used to form P_{t+1} and F is a set of fronts. The *nonDominatedSorting* function uses non-dominated sorting to produce a set of fronts from a list of individuals, the $assignCrowdingDistance$ function assigns crowding distance values to each solution in a given front and the $sortByCrowdingDistance$ function sorts a front in descending order using the crowding distance metric. $makeNewPop$ is a function which generates a new population by selecting parents using a binary tournament, performing user-defined crossover and mutation operators to create children. Solutions are compared in the tournament using the partial order relation $\prec_n$, defined as

Algorithm 4 NSGA-II

```
1: P_0 = generateRandomPopulation()
2: Q_0 = makeNewPop(P_0)
3: while termination criteria not satisfied do
4:     R_t = P_t ∪ Q_t
5:     F = nondominatedSorting(R_t)
6:     i = 0
7:     while |P_{t+1}| + |F_i| ≤ N do
8:         assignCrowdingDistance(F_i)
9:         P_{t+1} = P_{t+1} ∪ F_i
10:        i = i + 1
11:    end while
12:    sortByCrowdingDistance(F_i)
13:    for (j=0; j < N − |P_{t+1}|; j = j + 1) do
14:        P_{t+1} = P_{t+1} ∪ {F_i[j]}
15:    end for
16:    Q_{t+1} = makeNewPop(P_{t+1})
17:    t = t + 1
18: end while
```

$$i \prec_n j \; if(i_{rank} < j_{rank})$$
$$or((i_{rank} = j_{rank})$$
$$and(i_{distance} > j_{distance}))$$

where i and j are the solutions to be compared, i_{rank} is the index of the front containing solution i when non-dominated sorting is performed on the current population and $i_{distance}$ is the crowding distance of solution i.

3.3.2 SMPSO

Nebro et. al [78] propose the speed-constrained multi-objective PSO (SMPSO) algorithm as an improvement of previous multi-objective PSO work performed in [95]. SMPSO maintains a structure which contains non-dominated solutions, referred to as

an *archive*. At each iteration, particles deposit their current position into the archive if it is non-dominated with respect to all other archived solutions. In the case that the archive becomes full, the most crowded solution is removed. The crowdedness of solutions is determined using the crowding distance metric introduced in [28].

The defining characteristic of the SMPSO algorithm is the constriction of particle velocity. For this purpose a constriction coefficient X is used, derived from the constriction factor developed by Clerc and Kennedy [101] in Equation 2.4. Essentially, the constriction coefficient replaces the traditional upper and lower velocity bounds, capping the speed of each particle. The motivation behind this velocity constriction is the prevention of the swarm explosion phenomenon documented in [101].

SMPSO also bounds the accumulated velocity for each dimension j of particle i according to

$$v_{i,j}(t) = \begin{cases} \delta_j, & \text{if } v_{i,j}(t) > \delta_j. \\ -\delta_j, & \text{if } v_{i,j}(t) \leq \delta_j. \\ v_{i,j}(t), & \text{otherwise} \end{cases} \tag{3.2}$$

where $v_{i,j}(t)$ represents the velocity for dimension j of particle i at iteration t and $delta_j$ is calculated as

$$\delta_j = \frac{(UB_j - LB_j)}{2} \tag{3.3}$$

Note that in Equation 3.3, UB_j and LB_j represent the upper and lower bounds of the decision variable at index j, respectively. Summarizing the velocity constriction method, the velocity of a particle is calculated through the following steps:

1. Calculate particle velocity identically to traditional PSO using Equation 2.1.
2. Multiply the obtained velocity by the constriction factor calculated in Equation 3.2.

3. Constrain the resulting velocity using Equation 3.3.

Since identification of a definitively best non-dominated solution is not possible using only the dominance relation, selection of neighbourhood bests (also known as *leaders*) is non-trivial. To determine the neighbourhood best of a particle, two solutions are randomly selected from the leaders archive and the solution with the higher crowding distance is chosen. Personal bests are updated when a particle's current position dominates its personal best position.

SMPSO also employs mutation as a turbulence operator. Polynomial mutation [24] is applied to 15% of the particles in the swarm, selected randomly. This has the added benefit of helping to overcome local minima.

3.3.3 GDE3

One of the first multi-objective DE variants was the generalized differential evolution (GDE) algorithm. GDE extended DE by modifying the selection rule, where a trial position is accepted only if it weakly dominates the agents current position. One of the biggest flaws of this approach was the absence of a diversity maintenance mechanism. Despite this drawback, GDE has been shown to perform well but is extremely sensitive to the control parameters [60].

The GDE2 algorithm [61] proposed to improve the previous diversity flaws present in GDE. In the case that an agent's current position and trial position were non-dominating each other, the crowdedness of each was used to make a decision. This helped improve overall diversity of the population but decreased the potential to converge to the Pareto optimal front, since isolated non-dominated vectors far from the true front were given priority.

The third step of evolution for the GDE algorithms is the GDE3 algorithm, formally proposed in [62]. GDE3 improves the GDE2 algorithm by performing selection in the following manner:

- If the trial position weakly dominates the agent's current position, the trial position is accepted.
- If the current position of the agent dominates the trial position, the trial position is discarded.
- If neither position dominates the other, then the agent retains its current position and the trial position is added to the population as a new agent. Both positions now exist within the population.

Note that using these selection guidelines the population may have been increased up to size $2K$ after a generation, where K is the initial population size. If this is the case agents are then repeatedly removed from the population until the size is restored back to K. To determine which agents will be removed, agents are sorted into N fronts using a non-dominated sorting algorithm, e.g., [107]. Then, the crowding distance is calculated for each agent. Agents in the furthest front with the lowest crowding distance are removed. Essentially, this process ensures that dominated, crowded positions are given priority for removal.

3.4 WFG Benchmark Suite

Various suites exist for benchmarking the performance of MOOs. Outlined in [50], the general goals of any multi-objective benchmarking suite should be:

(a) Problems should not be susceptible to hill-climbing strategies.

(b) Unimodal test problems should be included. They are useful for testing convergence velocity relative to different Pareto optimal geometries and/or bias conditions.

(c) The test suite must include degenerate Pareto optimal fronts, disconnected Pareto optimal fronts and disconnected Pareto optimal sets.

(d) The majority of test problems in the suite should be multimodal.

(e) Several deceptive problems should be present.

(f) Problems should mainly be nonseparable.

(g) Several problems should be both nonseparable and multimodal to help simulate practical problems.

The WFG benchmarking suite, introduced by Huband *et al.* in [49], is one such suite which complies the above items. Within this work, the WFG suite is employed for algorithm benchmarking purposes due to the high level of customizability seen for each problem. Users are able to set the desired number of decision variables and desired number of objectives dynamically. Problems themselves within the WFG suite are defined in terms of a set of parameters. Each parameter set is derived through a series of transition vectors which each add complexity to the problem, helping simulate practical problem environments. Parameters of each problem are either distance-related or position-related. A formal definition [49] of a WFG problem is

$$
\begin{aligned}
\text{Given } z =& \{z_1, ..., z_k, z_{k+1}, ..., z_n\} \\
\text{Minimize } f_{m=1:n_c}(x) =& x_{n_c} + S_m h_m(x_1, ..., x_{n_c-1}) \\
\text{where } x =& \{x_1 ... x_{n_c}\} = \{\max(t^p_{n_c}, A_1)(t^p_1 - 0.5) + 0.5, ..., \\
& \max(t^p_{n_c}, A_{n_c-1})(t^p_{n_c-1} - 0.5) + 0.5, t^p_{n_c}\} \\
t^p =& \{t^p_1, ..., t^p_{n_c}\} \leftarrow\!\shortmid t^{p-1} \leftarrow\!\shortmid ... \leftarrow\!\shortmid t^1 \leftarrow\!\shortmid z_{[0,1]} \\
z_{[0,1]} =& \{z_{1,[0,1]}, ..., z_{n,[0,1]}\} = \{z_1/z_{1,\max}, ..., z_n/z_{n,\max}\}
\end{aligned}
$$

where $h_{1:n_c}$ are shape functions, $S_{1:n_c} > 0$ are scaling constants, x corresponds to a set of n_c parameters with domain [0,1], $A_{1:n_c-1} \in \{0, 1\}$ refer to degeneracy constants, z is a set of $k + l = n \geq n_c$ working parameters and $t^{1:p}$ are transition vectors where

"$\hookleftarrow$" is used to indicate that the transition vectors are formed from another vector using a transformation function. The domain of all $z_i \in$ z is [0, $z_{i,max}$] with each $z_{i,max} > 0$. A comprehensive overview of all functions in the WFG benchmarking suite can be viewed in [49].

Chapter 4

Many-objective Optimization

This chapter introduces the concept of many-objective optimization, covering pertinent background information. First, difficulties encountered when solving MaOPs are outlined. Next, a review of previous literature on many-objective optimization is given. All existing many-objective algorithms used within this work are detailed within the previous literature review.

4.1 Difficulties

Algorithms designed for MOO have continually demonstrated their superiority for solving problems with two or three objectives [28, 78, 106, 82, 62]. However, since many practical applications possess more than three objectives [36, 71, 77], it is highly desirable that MOO algorithms scale well when the number of objectives are increased. Unfortunately, MOO algorithms often perform radically different for MaOPs, e.g., [43]. Many CIMOOs have been shown to degrade as the number of objectives increase, with the Pareto-based optimizers possessing the overall worst scalability to MaOPs [52, 92].

While the failure of CIMOOs to scale to MaOPs is due to several reasons, the most significant is the vastly increased difficulty in attaining a uniformly spread set of solutions on the Pareto-optimal front. When the number of objectives is increased,

many CIMOOs experience difficulty when attempting to simultaneously maintain solution spread along with appropriate selection pressure to converge. As a result, many CIMOOs overprioritize either convergence or diversity, producing a well-spread set of solutions that are far from Pareto-optimal in the case of the former or a solution set which has converged to a small subregion near or on the Pareto-optimal front when the latter occurs. To illustrate the severity of this issue, consider that optimizers which only use the Pareto-dominance relation as selection criteria will nearly search randomly for MaOPs, since over 90% of a randomly generated initial solution set is non-dominated when the number of objectives are 8 or more [44]. Additionally, since most Pareto-based algorithms employ a secondary diversity-preservation metric alongside the dominance relation, solutions essentially spread rather than converge.

Maintenance of a well-spread set of solutions is another difficulty seen in MaOPs. Many CIMOOs employ the fast, computationally efficient crowding distance [28] operator, which yields satisfactory performance on two and three objective MOPs [59]. Unfortunately, the effectiveness of the crowding distance operator degrades considerably for MaOPs [59], largely due to the selection of dominance-resistant solutions [52], i.e., solutions with exceptional performance in one objective and extremely poor performance in many others. Since dominance-resistant solutions have a high chance of remaining non-dominated by definition, CIMOOs are essentially mislead by them. Using a diversity-preservation mechanism which is not susceptible to dominance-resistant solutions is key in maintaining a well-spread set of solutions for MaOPs. Another non-trivial issue regarding solution spread maintenance for MaOPs is the high computational complexity cost of accurately gauging the crowdedness of solutions. While this can be made faster by estimation methods, spread of the final approximated front may be impacted considerably.

The second major difficulty encountered in MaOPs concerns itself with the extremely large search space resulting from an increase in objectives. Consequently,

it is highly probable that many solutions found throughout the search are likely to be distant from each other. This has severe consequences for EMO, since when two distant parents are mated the offspring is likely to be far from both parents. Therefore it is necessary to employ some form of modification to recombination operators which ensures that offspring are reasonably close to parents. Without any modification, evolutionary optimizers will experience significantly reduced effectiveness of recombination operators, leading to difficulties during the search process.

Evaluating the performance of CIMOOs becomes increasingly more difficult as the number of objectives grow, another significant issue present for MaOPs. For many performance metrics, evaluating an approximation front produced by a CIMOO often requires considering a large number of high-dimensional points. Some metrics even become practically infeasible to calculate past a certain amount of objectives, such as the hypervolume metric. Since calculating exact hypervolume requires exponentially more computations with respect to the number of objectives [103], hypervolume values are often estimated using a Monte Carlo sampling technique [3] for MaOPs possessing more than ten objectives. This is problematic, since estimated values produce some degree of inaccuracy when comparing CIMOOs.

The final difficulty encountered for MaOPs relates to selection of a final solution from an approximated front produced by a CIMOO. Since the number of required approximation solutions increases exponentially with the number of objectives, an extremely large amount of solutions may be required to truly approximate the entire true Pareto front for MaOPs. Even if the true Pareto front is entirely captured, visualization of solutions in many-objective optimization is difficult in comparison to two or three objective MOPs. This presents a significant obstacle for a decision maker when selecting a final solution from the approximated front. Although this difficulty is not related to optimization directly, it presents significant obstacles for practical many-objective optimization.

The aforementioned difficulties of MaOps have presented a need for CIMOOs specifically designed to scale well when tasked with an increasing number of objectives. Unsurprisingly, recent years have seen a dramatic rise in the number of proposed many-objective CI algorithms. These algorithms can be largely separated into five distinct categories, each of which are described further in Section 4.2.

4.2 Literature Review

For MaOPs containing redundant objectives, dimensionality reduction techniques can be employed to shrink the number of objectives. Problems of this nature can be easily handled, since the classic multi-objective algorithms (see Section 3.3.1-3.3.3) can be used to tackle the reduced objective space, especially if the number of objectives becomes two or three. An example of this approach can be seen in [91], where the principal component analysis method is employed to remove redundant objectives.

MaOPs which do not possess a sufficient amount of redundant objectives present a considerably more difficult challenge, with a number of potential solutions proposed in previous literature. These approaches can be separated into five distinct categories, which are: Modification of the original dominance relation, usage of an additional convergence-related metric, usage of performance indicators, decomposition-based methods and non-Pareto approaches. Each category along with its relevant literature forms the rest of this section.

4.2.1 Dominance Relation Modification

Arguably the most straightforward way to adapt multi-objective optimizers for MaOPs is to modify the dominance relation in some way that promotes convergence to the Pareto-optimal front. The modified relations can also be considered as distinctly new relations which use the original Pareto-dominance concept as a base. The concept of

ϵ-dominance [64] was one of the first approaches to modify the dominance relation. Under the epsilon dominance relation, f is said to ϵ-dominate g for some $\epsilon > 0$ iff

$$(1 + \epsilon) \cdot f_i \geq g_i \tag{4.1}$$

where $i \in \{1, .., m\}$, m is the number of objectives and both f and g are feasible vectors in the decision space. The ϵ-dominance relation is essentially a relaxation of the original dominance relation, ignoring objectives which are only slightly worse. Choice of the ϵ value is problem specific. Typically, ϵ is determined by a decision maker and should be selected according to the physical meaning of the objective values.

Controlling dominance area of solutions (CDAS) [90] is another method which modifies the dominance region. CDAS proposes to alter the area dominated by a solution through adjustment of the angles that meet the axis bounding this area. Using an adjustable vector or parameters $\vec{S}$, CDAS shrinks the dominance area of objective i if $S_i < 0.5$ and grows the area if $S_i > 0.5$. This implies that setting $S_i = 0.5$ corresponds to using the regular Pareto dominance relation. The objective vector f of a solution $\vec{x}$ is transformed into f' by CDAS using the equation:

$$f'_i(\vec{x}) = \frac{r(x) \cdot sin(\lambda_i(x) + S_i\pi)}{sin(S_i\pi)} \tag{4.2}$$

$$\text{where } r(x) = ||f(x)|| \tag{4.3}$$

$$\text{and } \lambda_i = arccos(\frac{f_i(x)}{r(x)}) \tag{4.4}$$

CDAS has been used in place of the traditional dominance relation within the SMPSO algorithm (see Section 3.3.2) to form the CDAS-SMPSO algorithm [22], shown to perform well on a variety of MaOPs. However, a significant drawback

of the CDAS method is its extreme sensitivity to the choice of $\vec{S}$. Often, $\vec{S}$ must be empirically determined on a per-problem basis. Choosing an S_i that is too low results in the dominance relation becoming overly restrictive while an S_i that is too high results in low selection pressure towards the true Pareto front. A self-adaptive CDAS variant exists [89], however this adaptive variant has a runtime complexity of $O(n^2 - n)$ which is considerably greater than the $O(nlog(n))$ of the original CDAS, where n is the number of solutions to be compared.

The idea of fuzzy-based Pareto optimality, introduced in [44], applies the concept of fuzzy sets to determine the degree of domination between any two individuals. The fuzzy set is based on the left Gaussian function F_G [74], defined as

$$F_G(x) = \begin{cases} 1, & \text{if } x \leq c. \\ \exp(-0.5(\frac{x-c}{\rho})^2), & \text{otherwise} \end{cases} \tag{4.5}$$

where ρ defines the spread of the Gaussian function and c is a user-defined value, typically equal to the lower bound of x. The range of F_G is between zero and one, corresponding to the domination degree that one solution is better than another in one objective. The domain of F_G is between negative one and positive one, representing the difference between any two normalized individuals for a single objective. For each solution, normalization is performed by dividing each component j of its objective vector by the absolute value of the maximum difference among all solutions with respect to j.

Using F_G, the performance of an individual in comparison to another individual for an objective is a fuzzy set. Thus, m fuzzy sets exist when comparing individuals since there are m objectives. These fuzzy sets can then be used to determine the degree that one individual dominates another in the following way. Let p_i represent the performance of individual a in comparison to individual b on the i-th objective. Then, the domination degree σ^a corresponds to the truth degree of $p_0 \cap p_1 \cap ... \cap p_m$,

calculated as $p_0 \times p_1 \times ... \times p_m$.

From the definitions presented above, a is said to fuzzy-dominate b if $\sigma^a > \sigma^b$. Based on the method used to calculate σ, individuals are unlikely to possess the same σ value when compared. Thus, fuzzy Pareto dominance exerts more selection pressure than the original Pareto-dominance relation. As a result of this, fuzzy Pareto dominance is significantly more effective for converging towards the true Pareto front.

Since the fuzzy Pareto dominance relation returns a scalar dominance degree value, it is now possible to assign fitness to members based on their average dominance degree. Fitness assignment of an individual u begins by forming $N-1$ competitions, where N is the number of solutions which will be compared using the fuzzy Pareto dominance relation. For each competition between u and individual v in the competition pool, first obtain σ^u and $sigma^v$. Then, obtain the normalized performance of u with respect to v, calculated as $\sigma^u/(\sigma^u + \sigma^v)$. After performing all competitions of u, sum the normalized performances and divide by $N-1$. This value is then considered to be the fuzzy fitness of individual u.

Another method used to induce an ordering among non-dominated individuals is preference ordering (PO), originally introduced in [20]. An application of PO to many-objective optimization can be seen in [29], which proposed the PO_z and $PO_{z,k}$ algorithms. PO_k and $PO_{k,z}$ use the NSGA-II [28] algorithm as base, each sorting non-dominated individuals in a unique way. PO_k uses the concept of *efficiency of order*. A point p is efficient of order k if p is non-dominated with respect to all subsets of $f(p)$ having size k. That is, if p is not dominated by any other point in all of the $\binom{m}{k}$ subsets of its objectives then p has an efficiency of order k. Intuitively, if p is of efficiency m then p is not dominated by any other individual using the original Pareto dominance relation.

Several properties of order of efficiency are proven in [29]. Namely, if p is efficient of order k then it also efficient of order $k+1$. Additionally, if p is not efficient of

order k then it is not efficient of order $k+1$. Both of these properties are used in PO_k to rank the non-dominated individuals by their efficiency of order. Utilization of these properties also has the added benefit of improving runtime, since redundant comparisons are avoided.

The $PO_{k,z}$ algorithm adds a degree z to the efficiency of order definition. A point p is considered as efficient of order k with degree z if it is non-dominated for exactly z out of the possible $\binom{m}{k}$ k-element objective subsets. One should note that implementing a degree to the efficiency of order favors solutions which possess extreme components. This is not the case for the original efficiency of order definition.

4.2.2 Additional Convergence Metric

The use of an additional converge-related metric alongside Pareto dominance presents another potential solution for the failure of Pareto approaches in solving MaOPs. In this category, the traditional Pareto dominance relation is typically left intact and some form of secondary metric is used to distinguish which non-dominated solutions possess better convergence potential towards the true Pareto-optimal front. Diversity is either maintained through a separate crowdedness operator or indirectly by the convergence metric.

Köppen and Yoshida [58] propose several new convergence-related metrics which are used secondary to the original dominance relation. Each proposed method assigns solutions a *distance* value, which represents convergence potential in relation to all other solutions. All metrics assign distance by measuring the degree for which a solution A is dominated by a solution B using either a) the number of smaller or larger objectives, b) the magnitude of all smaller or larger objectives or c) some combination thereof. The subvector dominance (SV-DOM) method counts the number of objectives of A that are less desirable than the corresponding objectives in B. That is, for each objective i, if $f_i(A) > f_i(B)$ then the subvector dominance count

of A with respect to B is increased. The distance value of A is taken as the largest subvector dominance count observed when compared with all other solutions. In the case where A is dominated by at least one other solution, its distance value would be m. While the SV-DOM method is simple and easy to calculate, it does not account for the magnitude of dominance regarding each objective.

The eps-dominance (ϵ-DOM) method [58] takes into consideration all objectives of B that are worse than the respective objectives of A. Then, the *epsilon* value is computed, where ϵ denotes the minimum possible value that makes B strongly dominate A when subtracted from all objectives of B. That is, ϵ of A with respect to B is the minimum value such that $\forall i : f_i(A) < (f_i(B) - \epsilon)$. The distance value of A is taken as the smallest ϵ value observed when compared with all other solutions. Larger distance values of A would correspond to more "effort" needed to make other solutions dominate A, thus higher distance values are preferable. In the case where A is dominated by at least one other solution, its distance value would be 0.

A disadvantage of the ϵ-DOM method is that comparisons are essentially based on a single objective. Fuzzy Pareto dominance (FPD) seeks to rectify this by combining the magnitude of all worse objectives into a single scalar value. Therefore each objective is given equal consideration in determining solution desirability. FPD uses a bounded division operator BD defined as

$$BD(I_1, I_2) = \begin{cases} 1, & \text{if } I_1 > I_2. \\ I_1/I_2, & \text{otherwise} \end{cases} \tag{4.6}$$

where I_1, I_2 are numbers and $I_2 \neq 0$. The FPD method determines the dominance degree DD of A with respect to B using the following equation:

$$DD_{A,B} = \prod_{i=1}^{m} BD(f_i(A), f_i(B)) \tag{4.7}$$

Algorithm 5 SOD-CNT

```
1: for (j=0; j < N; j = j + 1) do
2:     U_j = ∅
3:     for (k=0; k < N; k = k + 1) do
4:         if k ≠ j then
5:             U_j = U_j ∪ (M_1(j, k), M_2(j, k))
6:         end if
7:     end for
8:     ND_j = non_dominated(U_j)
9:     for each z ∈ ND_j do
10:        z.distance = z.distance + 1
11:    end for
12: end for
```

Note that if B dominates A, $DD_{A,B} = 1$. The distance value of A is taken as the largest DD value observed when compared with all other solutions. Intuitively, smaller distance values are more desirable.

Sub-objective dominance count (SOD-CNT) [58] serves as a multi-criterion strategy which uses the strengths of two user-specified distance metrics. For a solution A, the set of solutions which "perform well" against A can be determined using the combination of both given distance metrics. The general process of SOD-CNT is described in Algorithm 5. Within Algorithm 5, M_1 and M_2 are distinct distance metrics and U_j is a set of 2-D distance vectors with each dimension d corresponding to the distance of solution j against another solution using M_d. The function *non_dominated* takes a set of distance vectors as input, returning the collection of solutions which correspond to the non-dominated vectors in the given set. The metrics chosen in [58] are SV-DOM and ϵ-DOM.

Grid-based fitness [66] is a recently proposed strategy which has been shown to be effective at increasing selection pressure towards the true Pareto front. Although grid-based fitness approaches do not directly use the standard non-dominance relation, non-dominance is indirectly promoted through the use of several convergence-related metrics. An evolutionary many-objective algorithm incorporating the concept of

grid-based fitness can be seen in [104]. Grid-based fitness approaches first form a grid of hypercubes within the search space. The lower and upper boundaries of the i-th objective are calculated as

$$LB_i = min_i - (max_i - min_i)/(2 \cdot div) \tag{4.8}$$

$$UB_i = max_i + (max_i - min_i)/(2 \cdot div) \tag{4.9}$$

where min_i and max_i are the minimum and maximum value of objective i respectively and div is a user-defined parameter which controls the number of divisions for each dimension. The hypercube width of the i-th objective is then formulated as

$$HW_i = (UB_i - LB_i)/div \tag{4.10}$$

Thus, each individual will reside within a distinct hypercube based on their position within the search space. To determine the hypercube co-ordinates of an individual y, each individual co-ordinate H_i can be found using

$$H_i(y) = \lfloor (F_i(y) - LB_i)/HW_i \rfloor \tag{4.11}$$

where F_i is the i-th objective value of individual y. Individuals are given a fitness value using three grid-based relations which together aim to achieve an ideal blend of convergence and diversity. These are the grid ranking (GR), grid crowding degree (GCD) and grid coordinate point distance (GCPD). Each relation is used to determine the desirability of each individual based on the hypercube that it is located in. GR serves as a measure of convergence, helping to distinguish individuals that are

converging best towards the true front. GR is simply defined as

$$GR(y) = \sum_{i=1}^{m} H_i(y) \tag{4.12}$$

A lower GR is desirable for minimization problems, since members of the true front will have low GRs. It is possible for solutions to be located in different hypercubes yet possess identical GRs. For example, consider solution A in hypercube (5,6) and solution B in hypercube (10,1). Both solutions would have a GR of 11 and thus need to be distinguished in another manner. For this purpose, the GCD metric is used to determine which solution is less crowded. While one could simply determine the crowdedness of an individual by counting the number of individuals within its hypercube, doing so would fail to account for the distribution of each solution. To remedy this problem, GCD also considers neighbouring hypercubes to give a better indication of crowdedness. GCD is calculated as

$$GCD(y) = card(NN(y)) \tag{4.13}$$

$$\text{where } NN(y) = \{z | z \neq y, \sum_{i=1}^{m} |H_k(y) - H_k(z)| < m\} \tag{4.14}$$

where $card(\cdot)$ denotes set cardinality and $NN(y)$ represents the nearest neighbours of individual y. There are several situations where individuals could have an equivalent GR and GCD, the most common being individuals within the same hypercube. Thus, the GCPD metric is adopted to establish a complete ordering among individuals, where smaller GCPD values correspond to more desirable individuals. GCPD is defined according to the following:

$$GCPD(y) = \sqrt{\sum_{i=1}^{m} (F_i(y) - (LB_i + H_i(y) \cdot HWi))} \tag{4.15}$$

4.2.3 Performance Indicator Usage

Integrating performance metrics during optimization is another valid approach for solving MaOPs. Performance indicators which combine diversity and convergence guide the search, promoting individuals which perform better for the selected indicator. The largest criticism of indicator-based optimizers is the high level of computational complexity required to calculate most useful indicators, e.g., hypervolume. Recent work [102, 3, 11] has focussed on reducing the computational complexity of indicators, however the issue is far from remedied for MaOPs.

While several multi-objective indicator based optimizers exist [112, 7, 51], their scalability to MaOPs is questionable as they were not explicitly designed for many-objective optimization. To the best of the authors knowledge only one many-objective indicator-based optimizer exists, which is the Hypervolume Estimation Algorithm for Multi-objective Optimization (HypE) [3]. HypE performs optimization based on the popular hypervolume indicator, which indicates convergence to the true front and solution spread in a single scalar value. However, since calculation of hypervolume is extremely expensive for MaOPs, approximate hypervolume values are generated when the number of objectives $m > 3$. The hypervolume-based fitness function f^H central to HypE is defined as

$$f^H(s) = \begin{cases} actual\_hypervolume(s), & \text{if } m \leq 3. \\ estimate\_hypervolume(s), & \text{otherwise} \end{cases} \tag{4.16}$$

where *estimate_hypervolume* estimates the hypervolume contribution of solution s using Monte Carlo sampling [2] and *actual_hypervolume* calculates the exact hypervolume contribution of s using an arbitrary hypervolume calculation method. Several examples of exact hypervolume calculation methods can be viewed in Section 6.1.1. Similar to a standard evolutionary algorithm, HypE performs the following actions until some pre-defined stopping criteria is reached:

1. **Mating Selection** - Using a binary tournament strategy, a set of parents Q is determined from the current population P. Comparison of two individuals is performed as follows:

   If $f_h(a) > f_h(b)$ **then** $Q = Q \cup a$

   Else $Q = Q \cup b$

2. **Variation** - User-defined mutation and crossover operators are applied to generate a set of N offspring from Q. The resultant offspring population is denoted R.

3. **Environmental Selection** - In an elitist fashion, the best N solutions are chosen from $R \cup Q$. To determine the best N solutions, non-dominated sorting is applied. Each front is added to the new population in ascending order until a front which only partially fits is encountered, denoted F_c. Fitness is calculated for each individual within F_c based on the f^H function and the best solutions in F_c are added into the new population until size N is reached.

4.2.4 Decomposition

Another promising avenue for solving MaOPs is the decomposition approach. Decomposition-based optimizers extract a set of subproblems from a given MOP, optimizing each of them simultaneously. The main representative of this category is the multi-objective evolutionary algorithm using decomposition (MOEA/D) algorithm, proposed in [106], which uses a user-defined aggregation function to decompose a MOP into a set of subproblems. Intuitively, a Pareto optimal solution to a MOP is an optimal solution of a scalar optimization problem corresponding to an aggregation of all objectives. The most common aggregation method is the weighted sum approach, detailed previously in Section 3.1. An alternative method is the Tchebycheff approach, which

converts a MOP into a single-objective problem using

$$\text{minimize } g^{te}(\vec{x}) = max\{\vec{w}_i|f_i(\vec{x}) - z_i^*|\} \tag{4.17}$$

where $i \in \{1, ..., n_c\}$, $\vec{w}$ is a weight vector, $\vec{z^*} = \{z_1^*, ..., z_m^*\}$ corresponds to the ideal point such that $z_i^* = min\{f_i(\vec{x})\}$ for all possible decision vectors $\vec{x}$. By modifying the choice of $\vec{w}$, different Pareto optimal points can be found. Thus, the entire Pareto-optimal front can be generated using this method. A downfall of this approach is that the aggregation function is not smooth for a continuous MOP. Another aggregation approach, which has been shown to perform well for MaOPs [27], is the penalty-based boundary intersection (PBI) method [106]. The PBI method defines a scalar subproblem as

$$\text{minimize } g^{pbi}(\vec{x}|\vec{w}, \vec{z^*}) = d_1 + \theta d_2 \tag{4.18}$$

$$\text{where } d1 = \frac{\left\|(\mathbf{F}(\vec{x}) - \vec{z^*})^T \vec{w}\right\|}{\|\vec{w}\|} \tag{4.19}$$

$$d2 = \left\|\mathbf{F}(\vec{x}) - (\vec{z^*} + d_1 \frac{\vec{w}}{\|\vec{w}\|})\right\| \tag{4.20}$$

where $\vec{x}$ is a feasible decision vector. To perform optimization, MOEA/D selects an arbitrary decomposition method and converts a MOP into R scalar optimization subproblems. For this purpose, K uniformly distributed weight vectors $\{w^0, ..., w^K\}$ are maintained each corresponding to a subproblem that will be optimized. A central idea to MOEA/D is that for subproblems x_i and x_j information about subproblem i should be helpful for solving subproblem j if w^i and w^j are similar, where $i, j \in K$.

MOEA/D utilizes the Pareto dominance relation to archive desirable solutions. For this purpose, an external structure NS which contains non-dominated solutions is employed. The initialization phase of MOEA/D begins by generating a random

initial population $x_1, ..., x_K$ and initializing $z = (z_1, ..., z_{n_c})$ using a problem-specific method. Next, for each i=1,...,R set $C_i = \{i_1,...i_R\}$, where $w^{i_1},..., w^{i_R}$ are the R weight vectors with the smallest Euclidean distance to w^i. The MOEA/D algorithm then performs the following steps for all K population members until some stopping criteria is satisfied:

Step 1) Reproduction: Select k, l $\in C_i$ and create a new solution q using x_k and x_l via user-defined crossover and mutation operators.

Step 2) Improvement: Optionally improve q using a problem-specific heuristic.

Step 3) Ideal Point Update: If $z_j < f_j(q)$ then $z_j = f_j(q)$, where $j \in \{1, ..., n_c\}$.

Step 4) Neighbouring Solution Update: For each $j \in C_i$, if $g(q, w^j) \leq g(x_j, w^j)$, then set $x_j = q$. Note that g is a function which takes a solution and weight vector as input, returning a scalar value indicating solution fitness.

Step 5) Archive Update: For each vector in NS, discard it if it is dominated by q. If no vector in NS dominates q, add q to NS.

The recently proposed NSGA-III algorithm [27] also uses decomposition-based ideas to solve MaOPs. The basic flow of NSGA-III is identical to its predecessor, NSGA-II (see Section 3.1.1), however diversity preservation is significantly different. At each generation, both NSGA-II and NSGA-III perform non-dominated sorting to partition the combined parent and offspring population into a set of fronts $F_0, F_1, ..., F_K$. Each front is then selected in ascending order, starting from F_0, to form the new population P_{t+1}. It is common for the final selected front, denoted F_L, to only fit partially into P_{t+1}. In this situation, some form of diversity score is calculated for each solution in F_L to determine which solutions should be inserted into P_{t+1}. It is here that NSGA-III differs from NSGA-II, as NSGA-III computes the diversity of each solution using a set of reference points rather than the traditional

crowding distance operator. Reference points can either be supplied by a user or generated in a structured manner. The approach seen in [27] uses Das and Dennis's boundary intersection approach [21], which places points on a normalized hyperplane possessing an intercept of one on each axis.

Regardless of the approach used to create reference points, the following process is performed in NSGA-III when computing the diversity of each solution in F_L: Let M_t denote $\cup_{i=0}^{L} F_i$. First, the ideal point $\vec{z} = (min(0), min(1), ...min(m))$ is constructed, where $min(i)$ denotes the minimum value of objective i in M_t. Objective values of all solutions in M_t are then modified, where objective value f_i becomes $f_i'(\vec{x}) = f_i(\vec{x}) - \vec{z}_i$. The newly modified objective values are then used to create m extreme vectors, where the extreme vector corresponding to objective i is a solution $x \in M_t$ minimizing a scalarizing function formed with a weight vector near the i-th objective axis. Next, the objective values of each solutions in M_t is normalized according to

$$f_i''(x) = \frac{f_i'(x)}{r_i} \tag{4.21}$$

where r_i represents the intercept of the i-th objective axis of the hyperplane formed using the m extreme vectors. Note that this normalization procedure is required to allow NSGA-III to handle problems possessing differently scaled objective values.

Now that each objective value has been normalized, solutions can be assigned to reference points. This is performed by joining each reference point with the origin to create a set of reference lines Z^r and assigning each member in M_t to the closest reference line in objective space. Thereafter, the niche count of each reference line is computed, where niche count σ_r is the number of members assigned to reference line r from $\cup_{i=0}^{L-1} F_i$. Lastly, niche counts are used to fill the remaining spots in P_{t+1}, detailed in Algorithm 6. Algorithm 6 utilizes the following functions:

- *least_assigned* - Takes a solution set S and set of reference points Z^R as input,

Algorithm 6 NSGA-III Niching Procedure

```
1: while |P_{t+1}| < K do
2:     r = least_assigned(P_{t+1}, Z^r)
3:     if σ_r = 0 then
4:         s = closest_assigned_solution(r, F_L)
5:     else
6:         s = random_assigned_solution(r, F_L)
7:     end if
8:     if s ≠ ∅ then
9:         P_{t+1} = P_{t+1} ∪ s
10:        σ_r = σ_r + 1
11:        F_L = F_L − s
12:    else
13:        Z^r = Z^r − r
14:    end if
15: end while
```

returning the reference line from Z^R possessing the least number of assigned individuals in S. In the case where multiple reference lines would be returned one of them is selected randomly.

- *closest_assigned_solution* - Takes a reference line r and solution set S as input, returning the closest solution to r out of all assigned solutions from S. If r has no assigned solutions in S, $\emptyset$ is returned.

- *random_assigned_solution* - Takes a reference line r and solution set S as input, returning a random solution from the set of solutions assigned to r in S. If r has no assigned solutions in S, $\emptyset$ is returned.

4.2.5 Non-Pareto Approaches

Abandoning the Pareto-dominance relation entirely is another viable possibility for scaling multi-objective optimizers to MaOPs. New relations would be used to determine the desirability of each solution, ideally incorporating a blend of convergence and diversity. If the relation favors convergence too much, diversity may be lost and

the search may converge to a small subregion of the Pareto-optimal front. Conversely, if the relation overvalues diversity it will encounter the same convergence issues of traditional Pareto dominance. Conformance to the original Pareto-dominance relation is also a desirable property, i.e., the new relation always considers non-dominated solutions as more desirable than non-dominated solutions.

Kukkonen and Lampinen [63] investigate several non-Pareto approaches based on the concept of *ranking dominance*. Ranking dominance orders solutions in terms of an aggregation function F_{agg}, which aggregates the ranking of a solutions individual objectives in some fashion. Two simple aggregation functions seen in [63] are F_{sum} and F_{min}. $F_{sum}(\vec{x})$ sums together the rank of each individual objective of $\vec{x}$, while $F_{min}(\vec{x})$ returns the minimal ranking out of all objectives for $\vec{x}$. The F_{min} and F_{sum} method are formally defined as

$$F_{sum}(\vec{x}) = \sum_{i=1}^{m} rank(i, f_i(\vec{x})) \tag{4.22}$$

$$F_{min}(\vec{x}) = \min_{i=1..m} rank(i, f_i(\vec{x})) \tag{4.23}$$

where $\vec{x}$ is a feasible decision vector and $rank$ is a function which takes an objective index i and objective value $f_i(\vec{x})$ as input, returning the rank of $f_i(\vec{x})$ when compared to all other objective vectors in the current front using objective i. An example of the process used to calculate F_{sum} and F_{min} can be viewed in Figure 4.1. Note that the F_{sum} method has shown mixed results empirically, as a comparative study by Corne and Knowles [18] demonstrates competitive performance on practical problems, yet poor performance is exhibited on a popular benchmarking suite in work by Garza-Fabre *et al.* [37].

An obvious fault of F_{sum} and F_{min} is that they do not take into account the distribution of each objective. The sum of ratios (SR) method [6] is one such rank

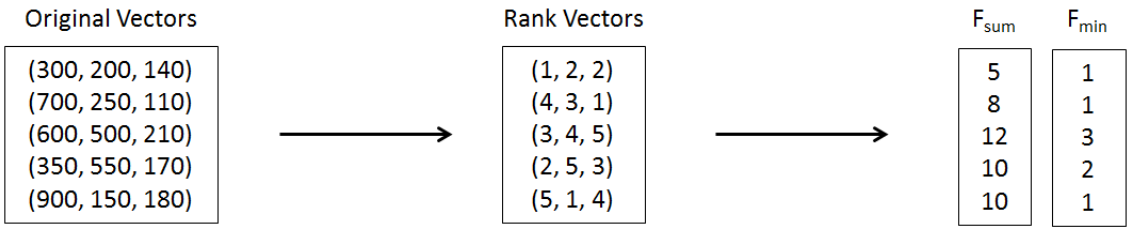

Figure 4.1: Calculation of the F_{sum} and F_{min} aggregation methods are demonstrated for a front containing five solutions. The MOP in question possesses three objectives.

dominance technique which considers the distribution of objective values. SR normalizes using the range of values of each objective using the equation

$$SR(\vec{x}) = \sum_{i=1}^{m} \frac{f_i(\vec{x}) - min(i)}{max(i) - min(i)} \tag{4.24}$$

where max and min are functions returning the maximum and minimum values of objective i in the solution set, respectively. Another potential improvement of the F_{sum} method is to consider the range of rankings for each objective, normalizing via the equation

$$NF_{sum}(\vec{x}) = \sum_{i=1}^{m} \frac{rank(i, f_i(\vec{x}))}{maxRank(i)} \tag{4.25}$$

where NF_{sum} denotes the normalized F_{sum} method and $maxRank(i)$ returns the maximum rank observed for objective i. Previous work by Ross et. al [4, 88] demonstrates the NF_{sum} method to perform promisingly.

Since F_{sum}, NF_{sum} and F_{min} each prioritize convergence, diversity must be maintained using an external metric. Unfortunately this is non-trivial, since solutions are already ordered according to the chosen aggregation method. One method of preserving diversity seen in [63] is to gradually increase the number of solutions selected according to a diversity metric in a linear fashion. This method was incorporated into the GDE3 algorithm [62] (see Section 3.3.3) with the NF_{sum} method, linearly increasing the number of solutions selected according to diversity such that:

- At generation 0, selection would be done according to the ordering imposed by the F_{sum} method.
- Let G_{max} denote the maximum number of generations. At generation $G_{max}/2$, half of selection would be done based on the F_{sum} ordering and the other half would be performed according to a diversity-preservation operator.
- At generation G_{max}, all selected is performed using a diversity-preservation operator.

The motivation behind this method is to promote convergence early on and prioritize diversity in later stages of the search when the population has already converged. The L-dominance relation [116] is another non-Pareto approach which induces an ordering among solutions. Before defining the concept of L-optimality, several preliminary definitions must first be established.

Definition 1 - A normalized MOP (NMOP) has the same formulation as a MOP (see Section 3.1), except each f_i is normalized using

$$f_i'(\vec{x}) = \frac{f_i(\vec{x}) - min(i)}{max(i) - min(i)} \tag{4.26}$$

Thus, for an arbitrary NMOP each normalized objective is in the range [0,1].

Definition 2 - The p-norm of a decision vector $\vec{x}$ is the distance to the utopian point (the origin) of the NMOP, derived as follows:

$$||N(\vec{x})||_p = \frac{\sum_{i=1}^{m} f_i(\vec{x})^p}{p} \tag{4.27}$$

Definition 3 - Let B_t be a function which takes decision vectors $\vec{x_1}$ and $\vec{x_2}$ as input and returns a count of the number of objectives for which $\vec{x_1}$ is better than $\vec{x_2}$. A formal definition of B_t is

$$B_t(\vec{x_1}, \vec{x_2}) = |\{i \in \mathbb{N} | i \leq m \wedge f_i(\vec{x_1}) < f_i(\vec{x_2})\}|$$

We are now ready to define the concept of L-dominance. A decision vector $\vec{x_1}$ is considered to L-dominate $\vec{x_2}$ if, and only if, the following two conditions are satisfied:

1. $B_t(\vec{x_1}, \vec{x_2}) - B_t(\vec{x_2}, \vec{x_1}) > 0$

2. $||N(\vec{x_1})||_p < ||N(\vec{x_2})||_p$ (for a user-defined p value)

It is shown in [116] that Pareto-domination implies L-domination, but L-domination does not imply Pareto domination. Additionally, if a solution is L-optimal then it is also Pareto optimal. One should note that in the case where $m = 2$, L-optimality is equivalent to Pareto optimality.

Chapter 5

Proposed Approaches

This chapter discusses each of the proposed many-objective optimization algorithms. Implementation-level details along with a comprehensive description is given for each algorithm.

5.1 Knee-Driven Algorithms

We propose several new many-objective optimizers incorporating the idea of knee points. Within the field of multi-objective optimization, knee points are a subset of Pareto-optimal points for which an improvement in one objective results in severe degradation of at least one other objective. Knee points represent the naturally preferable points within the Pareto optimal front when no problem-specific knowledge is available, as they possess maximal marginal rates of return. Since each objective is regarded as equally important in the absence of a decision maker, minimally improving an objective while immensely degrading others often cannot be justified. An illustration of a knee point for a bi-objective minimization MOP can be viewed in Figure 5.1. Within Figure 5.1, Solutions A-E form the non-dominated front, each displayed with a gray fill. Here, solution C is a knee point due to its larger marginal rates of return in comparison to the other non-dominated solutions. Notice that solution $C = (3, 3)$ has its nearest non-dominated neighbours in objective space at $B = (2, 8)$

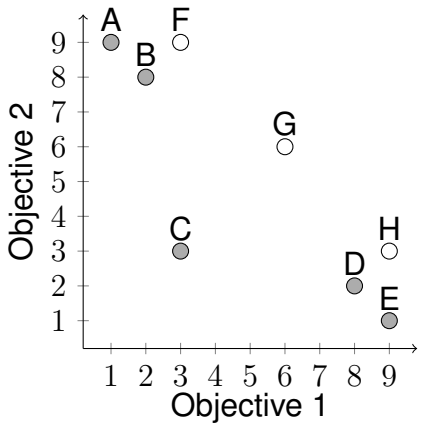

Figure 5.1: Eight solutions are shown for a bi-objective MOP. Each non-dominated solution is displayed with a gray fill. Here, solution C is a knee point due to its large marginal rates of return in comparison to the other non-dominated solutions.

and $D = (8, 2)$. Slightly improving one objective of C requires a much larger sacrifice in the other objective.

Since knee points are a subset of non-dominated points, they can be used to further distinguish the desirability of non-dominated solutions. This approach would fall under the additional convergence metric category (Section 4.2.2). Knee points within the current front are given archival priority and the search is guided around them. It is important to realize that solutions identified as knee points during the search are often not true knee points of the Pareto-optimal front, since the current front is simply an approximation. Rather, knee points of the current front represent solutions that are converging best within their immediate neighbourhood and are therefore useful for increasing the selection pressure to converge towards the Pareto-optimal front. This can be also be considered as a bias towards a higher hypervolume value. As the hypervolume metric is maximized if, and only if, the solution set consists entirely of all Pareto-optimal points [113], utilizing knee points via this method should theoretically help an algorithm converge to the true Pareto front.

To the best of the authors knowledge, the knee-driven evolutionary algorithm (KnEA) [108] is the only existing MOO which incorporates knee points during op-

timization. KnEA uses the NSGA-II (see Section 3.3.1) algorithm as base, incorporating knee points in several ways. When performing a binary tournament between two mutually non-dominating solutions during the selection phase, if either solution is a knee point it is regarded as more desirable and wins the tournament. If neither solutions (or both) are knee points, the solution with the higher weighted distance metric WD is used to promote diversity, where WD of a solution x is calculated using

$$WD(x) = \sum_{i=1}^{k} w(x^i) \cdot d(x, x^i) \tag{5.1}$$

$$w(x^i) = \frac{rank(x^i)}{\sum_{j=1}^{k} rank(x^j)} \tag{5.2}$$

$$rank(x^i) = \frac{1}{|d(x, x^i) - \frac{1}{k}\sum_{j=1}^{k} d(x, x^j)|} \tag{5.3}$$

where k is the desired number of nearest neighbours to consider, x^i is the i-th nearest neighbour of x in objective space, $d(x, y)$ is the Euclidean distance between solutions x and y in objective space, $w(x^i)$ represents the weight of the i-th nearest neighbour of x and $rank(x)$ corresponds to the rank of $d(x, x^i)$ in comparison to all nearest neighbour distances $d(x, x^i)$, $1 \leq j \leq k$. Knee points are also utilized during the environmental selection phase as KnEA uses knee points as a secondary selection criterion rather than the traditional crowding distance of NSGA-II. This method essentially promotes non-crowded knee points during the search.

Inspired by KnEA, we apply knee points to both the SMPSO [78] and GDE3 [62] algorithms in a similar manner. The resulting algorithms are referred to as knee-driven PSO (KnPSO) and knee-driven DE (KnDE), respectively.

5.1.1 Knee Point Identification

The knee point identification mechanism used by KnPSO and KnDE is identical to the approach used in KnEA. First, a list, E, of extremal solutions is determined, where an extremal solution E_i is defined as the solution with the least desirable fitness for objective i. Next, an extremal hyperplane H is constructed as follows:

1. Create all possible $n_c - 1$ distinct vectors using the solutions of E.
2. Construct a matrix M, where each row of M is a vector computed in 1).
3. Compute the null space of M, using the result to determine the constants of the plane equation of H.

H can then be used to determine the set of knee points of the approximated front using the following method: For each candidate solution s, s is deemed a knee point if, and only if, s possesses the maximum objective space distance to H within its neighbourhood. An adaptive strategy [108] is used to determine the neighbourhood size of each solution. The neighbourhood of a solution is a hypercube with n_c sides, where each side is calculated as

$$R_t^j = (max_t^j - min_t^j) \cdot r_t \tag{5.4}$$

where max_t^j and min_t^j denote the maximal and minimal values of the j-th objective at time t respectively and r_t corresponds to the ratio of the neighbourhood size to the range spanned by objective j at time t. r_t is updated using

$$r_t = r_{t-1} \cdot e^{-\frac{1-(\alpha_{t-1}/T)}{n_c}} \tag{5.5}$$

where α_{t-1} denotes the ratio of knee points to non-dominated solutions at time $t-1$ and T is a user-defined parameter which represents the desired ratio of knee points

to non-dominated solutions, with $0 < T < 1$. Note that initially $\alpha_t = 0$ and $r_t = 1$. The strategy above essentially shrinks and grows the neighbourhood size adaptively until the ratio of knee points to non-dominated solutions in the solution set converges to T.

Given that we have described the exact mechanism used to determine knee points, we can now define several knee-based comparison operators as follows:

$$i \prec_k j \; if \; (i_{rank} < j_{rank}) \; or \; (i_{rank} = j_{rank} \; and \; (isKnee(i) \; and \; !isKnee(j))) \quad (5.6)$$

$$i \prec_d j \; if \; (i \prec_k j) \; or \; (WD(i) > WD(j)) \quad (5.7)$$

$$i \prec_c j \; if \; (i \prec_k j) \; or \; (E_D(i) > E_D(j)) \quad (5.8)$$

where $isKnee$ is a function which determines if a given solution is a knee point using the identification mechanism described in this section, i_{rank} denotes the index of the front containing solution i when non-dominated sorting is performed on the entire set of solutions and $E_D(x)$ is the objective space distance of solution x to the extremal hyperplane H. Note that $\prec_k$ is an operator which deems knee points most desirable, $\prec_d$ represents a diversity operator which prefers non-crowded knee points and $\prec_c$ is a convergence operator which prefers converging knee points.

5.1.2 KnPSO

The general flow of KnPSO is identical to the SMPSO algorithm, described in Section 3.3.2. Similar to SMPSO, KnPSO maintains an archive of non-dominated solutions. However, KnPSO differs considerably from SMPSO with regards to the archive maintenance phase. In traditional SMPSO, when the maximum size of the archive is surpassed the worst (most crowded) solution is removed using the crowding distance [28] metric. Since KnPSO utilizes knee point status as the primary solution desir-

Algorithm 7 KnPSO Archive Maintenance

1: $Q = A \cup S$
2: $set\_dominance\_ranks(Q)$
3: $H = extremal\_hyperplane(Q)$
4: $determine\_knees(Q, H)$
5: $R = sort(\prec_d, Q)$
6: $A = \emptyset$
7: **for** $(i = 1; i \leq q; i = i + 1)$ **do**
8: $\quad A = A \cup R_i$
9: **end for**

ability criterion, the set of the knee points would have to be recalculated after each archive removal if the removal method of SMPSO was used without modification. This would require a considerable amount of computational effort, since the extremal hyperplane would have to be recreated and the distance of each solution would have to be recalculated every time a solution was added to the archive.

KnPSO avoids this time-consuming process by adding all solutions to the archive and then returning the archive to its maximum allowed size, requiring the set of knee points to only be determined once. This archive maintenance process is shown in Algorithm 7. The archive solutions are determined as follows: First, combine the current archive A and the current swarm S as $Q = A \cup S$. Next, perform efficient non-dominated sorting (ENS) [107] on Q to divide Q into non-dominated fronts $F_0, ..., F_N$, setting the dominance rank of each solution to be its non-dominated front index. The extremal hyperplane H is then constructed using the extremal solutions of Q and the knee points of Q are determined using H. Finally, Q is sorted according to the $\prec_d$ operator and the top q solutions of Q form the new archive, where q is the maximum archive size. In the case where $\prec_d$ fails to distinguish between two solutions, one of the compared solutions is selected randomly. Since the $\prec_d$ operator prefers converging knee points, and the knee point identification mechanism outlined in Section 5.1 implicitly considers solution diversity, the archive should contain a good blend of both convergence and solution spread.

Selection of neighbourhood bests is another considerable difference of KnPSO. KnPSO determines the neighbourhood best of each particle by choosing two random solutions in the archive and selecting whichever one is deemed more desirable. Solution desirability is determined by the $\prec_c$ operator. In the unlikely event where $\prec_c$ fails to distinguish between two solutions, one of them is selected randomly. Essentially, KnPSO prefers leaders to be non-crowded knee points.

5.1.3 KnDE

KnDE utilizes the traditional DE/rand/1/bin flow (see Section 2.3) where a trial position is created for each agent of the population P at each generation according to Algorithm 2. When comparing a trial position $\vec{t}$ to an agent's current position $\vec{x}$, the following selection scheme is employed:

- If $\vec{t}$ dominates $\vec{x}$, accept $\vec{t}$ as the agent's new position
- If $\vec{x}$ dominates $\vec{t}$, discard $\vec{t}$
- If $\vec{t}$ and $\vec{x}$ are non-dominated with respect to each other, then the agent retains its current position and a new agent with position $\vec{t}$ is added to P

Note that at each generation, the population may exceed the maximum allowed size K due to the addition of non-dominated trial positions in accordance with the above selection guidelines. In this case, a removal procedure is initiated to shrink the population back to size K. KnDE performs population size reduction in a manner similar to the archive maintenance method of KnPSO, as P is pruned according to a knee-based operator and the top K solutions are retained, promoting both convergence and solution spread implicitly. This process is detailed in Algorithm 8. Note that within Algorithm 8 $index\_of\_worst$ is a function which accepts a comparison operator and solution set as input, returning the index of the worst solution with

Algorithm 8 KnDE Population Reduction

```
1: set_dominance_ranks(P)
2: H = extremal_hyperplane(P)
3: determine_knees(P, H)
4: while |P| > K do
5:     j = index_of_worst(≺_d, P)
6:     P = P − P_j
7: end while
```

respect to the given operator. Within this work the $\prec_d$ operator is used to determine the worst solution, however the $\prec_c$ operator can be employed instead to prioritize convergence to the true front rather than solution spread. Regardless of the comparison operator chosen, both convergence and diversity will be considered to some degree due to the knee point identification mechanism employed in KnDE, described in Section 5.1.

Note that when comparing two solutions, the $\prec_d$ operator will fail to distinguish solutions in the unlikely event where each solution has an identical distance to H. In this situation, one of the solutions is randomly selected.

5.2 SrEA/D

The third algorithm proposed in this thesis is the sum-of-ranks evolutionary algorithm using decomposition (SrEA/D). SrEA/D serves as a hybrid algorithm, combining the promising decomposition approach [65, 69] with the NF_{sum} relation [6, 18]. For an overview of these methods, refer to Sections 4.2.4 and 4.2.5, respectively. The core concept of SrEA/D is to utilize the powerful convergence properties of the NF_{sum} method to draw solutions closer to the Pareto-optimal front while simultaneously promoting an equidistant spread of solutions indirectly using uniformly spaced reference points. Rather than explicitly utilize a spread preservation metric, SrEA/D maintains a well-spread solution set by dividing the objective space into subregions using its generated reference points. Specifically, subregion ψ^i is defined as

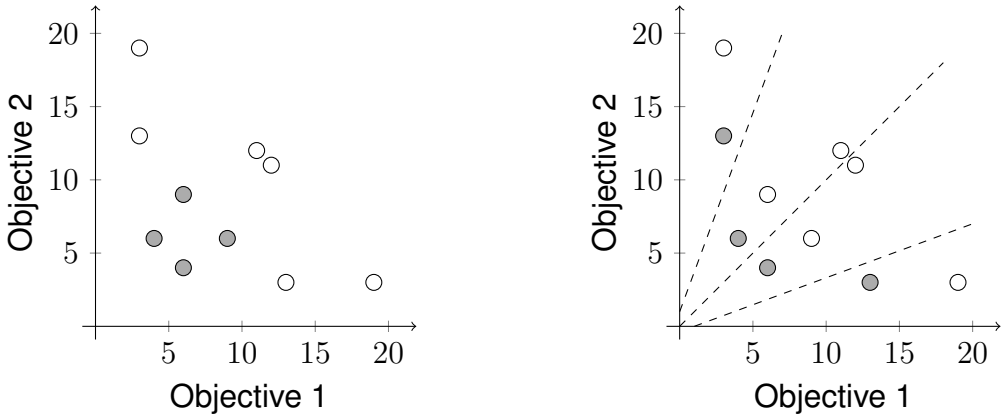

Figure 5.2: The spread-preservation advantages of the subregion concept is exemplified for a population of ten solutions. The four best solutions according to only the NF_{sum} relation are shown on the left, whereas the four most desirable solutions according to SrEA/D, which employs NF_{sum} in each individual subregion, is shown on the right. The selected best solutions are displayed with a gray fill.

$$\psi^i = \{\vec{v} \in R^m | \forall j \in \{1, ..., K\} : \langle \vec{v}, \vec{w}_i \rangle \leq \langle \vec{v}, \vec{w}_j \rangle\} \tag{5.9}$$

where $i \in \{1, .., K\}$, x is a feasible decision vector and $\langle \vec{v}, \vec{w}_i \rangle$ corresponds to the acute angle between $\vec{v}$ and $\vec{w}_i$. Essentially, SrEA/D partitions the objective space into K subregions, where K is the desired population size. The motivation behind this partitioning is to prevent the loss of diversity experienced by the traditional NF_{sum} method, since we can now employ NF_{sum} in each subregion rather than the entire population to better maintain solution spread. Figure 5.2 exemplifies how the subregion partitioning of SrEA/D maintains a better spread set of solutions in comparison to using the NF_{sum} relation alone. Shown on the left, when selecting the four most desirable solutions using only NF_{sum}, the diversity of the solution set is sacrificed for convergence. Here, the population would be led into a small region of the Pareto-optimal front. However, when partitioning the objective space into four subregions and selecting the best converging particle from each subregion using NF_{sum}, shown on the right of Figure 5.2, the selected solutions are simultaneously converging and well-spread.

Algorithm 9 SrEA/D Basic Framework

1: $initialize()$
2: **while** termination criterion is not satisfied **do**
3: $\quad Q = mating\_selection()$
4: $\quad R = create\_offspring\_population(Q)$
5: $\quad S = P \cup Q$
6: $\quad P = truncate\_population(S)$
7: **end while**

Note that if the final population consists entirely of Pareto-optimal solutions in unique subregions, SrEA/D will have produced a set of uniformly-spaced Pareto-optimal points for any uniform MOP. This is a very useful property, since an equidistant set of Pareto-optimal solutions is ideal for a decision maker to select from. The basic framework of SrEA/D is shown in Algorithm 9. Until some termination criterion is satisfied, the following steps are performed:

Step 1) Initialization: Initialize the population P randomly. Generate a set of $R = \{r^1, ..., r^T\}$ uniformly spaced reference points from a unit simplex using Das and Dennis's systematic boundary intersection method [21], where $T = \binom{D+n_c-1}{n_c-1}$ and D is the number of desired divisions. In the case where $n_c \geq 7$, both a boundary layer and inner layer are utilized to reduce the total number of generated reference points. The components of reference points in the inner layer are resized using

$$i_j = \frac{1-\phi}{n_c} + \phi \times i_j \tag{5.10}$$

where $\vec{i}$ is a reference point in the inner layer, $j \in \{1, ..., n_c\}$ and $\phi \in [0, 1]$. In this thesis, we set $\phi = 0.5$ in all cases.

Step 2) Mating Selection: The mating selection process of SrEA/D is shown in Algorithm 10. Here, a set of parents Q is determined for mating purposes. For each reference point r^i, a subregion ψ^j is selected from the closest T subregions to $\vec{r}_i$. Thereafter, we perform a binary tournament to select a parent solution from ψ^j, using the NF_{sum} relation to determine solution desirability. In other words, the solution

Algorithm 10 SrEA/D Mating Selection

```
1: Q = ∅
2: for each r^i ∈ R do
3:     Randomly choose a subregion ψ^j from the T closest subregions to R^i
4:     if |ψ^j| = 0 or rand() > δ then
5:         Add a random solution in P to Q
6:     else if |ψ^j| = 1 then
7:         Add the lone solution residing in ψ^j to Q
8:     else
9:         x' = binary_tournament(ψ^j)
10:        Add x' to Q
11:    end if
12: end for
13: return Q
```

with the lowest sum of objective ranks when ranked against all other solutions in ψ^j wins the tournament. Note that to benefit population diversity, a small probability (1-δ) to select parents from the entire population is included. In the case where no solutions exist in the closest T subregions, Q is selected randomly from the entire population by default.

Step 3) Offspring creation: Here, a set R of K offspring are generated by applying recombination operators to each pair of parent in Q. ln theory, any crossover and mutation operators can be used. In this work, we employ the simulated binary crossover [25] and polynomial mutation [26] operators.

Step 4) Population Truncation: The final step is to truncate the combined population $S = P \cup R$ by continually removing undesirable solutions until $|S| = K$. Algorithm 11 presents the entire population truncation process employed in SrEA/D. When determining a solution to remove, the subregion that is most crowded, i.e., contains the most solutions, is first selected as ψ^k. Next, the *set_ranks* method is executed on ψ^k, assigning a sum-of-ranks value to each $x' \in \psi^k$ using the NF_{sum} relation. Note that the NF_{sum} method only ranks solutions against other solutions residing in ψ^k rather than the entire population. Lastly, we remove from S the

Algorithm 11 SrEA/D Population Truncation

1: **Inputs:** Combined population S
2: **while** $|S| > K$ **do**
3: Get the most crowded subregion ψ^k in P. Ties are broken randomly.
4: $set\_ranks(\psi^k)$
5: Identify y as the solution in ψ^k possessing the largest F_{sum} value.
6: $S = S - y$
7: **end while**
8: **return** S

solution that is converging worst in ψ^k, i.e, the solution with the worst F_{sum} value. Since the solution that is converging worst in the most crowded subregion is always selected for removal during the truncation phase, SrEA/D is an elitist algorithm.

Chapter 6

Experimental Setup

This chapter details the experimental setup used within this thesis. Section 6.1 presents the various performance measures used for algorithm comparison. In Section 6.2, an overview of the statistical methodology employed within this work is covered. Various functions used for benchmarking purposes are given in Section 6.3. Lastly, Section 6.4 presents the parameter sets used for all algorithms.

Experiments were performed via a highly customized version of the jMetal framework [31]. All pseudo-random number generation was performed via an implementation of the Mersenne Twister [73].

6.1 Performance Measures

Measuring the performance of multi-objective optimization algorithms is a non-trivial task, since the Pareto-optimal front is often not known. Many performance measures tend to analyze the obtained approximation front using various criteria. These criteria can be categorized as follows:

- *Distribution* - the overall distribution of the non-dominated solution set
- *Variance* - non-dominated solution spread with respect to the minimum/maximum objective values

- *Diversity* - amount of different non-dominated solutions
- *Precision* - similarity of the approximation front to a best known estimation

The performance measures used in this work incorporate each of the above criteria to help give an accurate, unbiased indication of how well a MOO has performed.

6.1.1 Hypervolume Indicator

The hypervolume indicator [114] is a metric which simultaneously measures both the spread of solutions along with the level of convergence to the Pareto-optimal front. The hypervolume metric is represented as a single scalar value, calculated as the sum of sub-cuboids created by solution points. An extensive overview of various hypervolume calculation algorithms can be found in [8]. An important property of the hypervolume metric is that it is maximized if, and only if, the solution set consists entirely of all Pareto-optimal points [35]. Therefore, hypervolume can be used to give a good indication of how close a MOO has come to producing a set of solutions on the true Pareto-optimal front. Another favorable property of the hypervolume metric is that it is capable of proving that a solution set is not worse than some other solution set for all solution pairs [115].

A considerable drawback of the hypervolume metric is its high computational complexity as an NP-hard problem [12], as naive calculation takes exponential time with respect to the number of objectives. This is a serious drawback, as it becomes infeasible to calculate the exact hypervolume metric value for problems possessing a sufficiently large number of objectives. Consequently, for MaOPs it is common for hypervolume to be estimated using Monte Carlo sampling [3] rather than calculated using an exact method. However, recently a fast method for calculating exact hypervolume has been proposed as the WFG algorithm [102]. This method works as follows: Let $\vec{r}$ denote the objective values of an arbitrary point p. WFG utilizes the

observation that the exclusive hypervolume of p relative to a set S is equivalent to the difference between the inclusive hypervolume of p and the hypervolume of S with each point limited by $\vec{r}$. The WFG algorithm applies this technique to each member of S, efficiently calculating the hypervolume metric.

To calculate hypervolume within this work, each objective value is normalized to the range [0,1], where 0 and 1 represent the best and worst objective values in the current population, respectively. The reference point used for hypervolume calculation purposes is chosen to be the nadir point, i.e a vector where the value at index i contains the worst value of objective i, where $i \in \{1, ..., n_c\}$. Note that since each objective vector is normalized, the nadir point is the unit vector of length n_c.

6.1.2 Inverted Generation Distance

The inverted generational distance (IGD) [39] is a metric which measures convergence of the approximated front POF^* towards the true Pareto optimal front POF'. IGD is calculated via the following equation:

$$IGD = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n} \tag{6.1}$$

In the equation above, n represents the number of solutions in POF' and d_i is the Euclidean distance in the objective space between solution i of $POF'$and the closest member of POF^*. Minimization of this metric corresponds to closer convergence towards the true Pareto optimal front. One should note that when $IGD = 0$, $POF^* = POF'$.

IGD conforms to the principle of monotony, since it considers a POF^* with a variety of non-dominated solutions that are close to POF' to be more desirable than a POF^* with a single solution on POF'. IGD was introduced as a result of its monotony property, since the original generational distance metric did not adhere

to this principle. A drawback of IGD is that it becomes computationally expensive to calculate for a large POF', since for each solution in POF' it must evaluate the distance to every solution in POF^* to determine the nearest neighbour.

6.2 Statistical Methods

Statistical methods are divided into two main categories according to what is known about the population at hand. The popular *parametric* category consists of tests which make assumptions about the underlying distribution, often assuming that the distribution is normal. Conversely, *non-parametric* tests do not rely on the assumption that data has been drawn from a given distribution. Thus, these methods are considered to be *distribution free* since they have no dependence on the given population. Since normal variances cannot be assumed[1] for a stochastic optimizer [57], the methods used in this work are non-parametric. This helps to ensure that type I and II errors are reduced as much as possible since the data is not guaranteed to be normally distributed [19].

The Mann-Whitney-Wilcoxon rank sum test [72], described in Section 6.2.1, was used in a pairwise fashion to test for statistical significance in all experiments. Each Mann-Whitney-Wilcoxon rank sum test was performed with a 95% confidence level. If a statistically significant difference between a pair of algorithms existed, the algorithm with the higher mean was given a point and the algorithm with the lower mean was deducted a point. All experiments reference *difference*, which is simply the difference between pairwise wins and losses for each algorithm. Additionally, the *rank* of each algorithm denotes the algorithm ranking in comparison to all other algorithms with respect to the number of pairwise wins for a given benchmark function. All experiments were recorded using 30 runs.

[1] However, it is still possible for a stochastic optimizer to possess a normal distribution

6.2.1 Mann-Whitney-Wilcoxon Rank Sum Test

The Mann-Whitney-Wilcoxon rank sum test [72], also known as the Mann-Whitney U-test, is used to determine whether two independent samples of observations are drawn from identical distributions. The test is built around the idea that arranging random variables together in increasing order of magnitude will help draw conclusions about the relationship between the parent populations. If sufficient evidence of random mixing is not present, the null hypothesis of identical distribution is rejected. The Mann-Whitney-Wilcoxon rank sum test makes a few assumptions beforehand:

- The two samples are random.
- The two samples are independent of each other.
- Observations are arranged by rank.

To calculate this test. first find the individual sum of the ranks R_1 and R_2. Next, determine the U statistic for each sample using the following equations:

$$U_1 = n_1 n_2 + \frac{n_1(n_1+1)}{2} - R_1 \tag{6.2a}$$

$$U_2 = n_2 n_1 + \frac{n_2(n_2+1)}{2} - R_2 \tag{6.2b}$$

$$U = min\{U_1, U_2\} \tag{6.2c}$$

where n_1 and n_2 represent the sizes of sample 1 and 2, respectively. A z score can then be calculated as:

$$\mu = \frac{n_1 n_2}{2} \tag{6.3a}$$

$$\sigma = \sqrt{\frac{n_1 n_2 (n_1 + n_2 + 10)}{12}} \tag{6.3b}$$

$$z = \frac{U - \mu}{\sigma} \tag{6.3c}$$

The z score calculated above is then compared with a critical value associated with a pre-determined confidence level, α. The null hypothesis which states that the two populations have identical distribution functions is then either rejected or accepted based on this comparison. One should note that the Mann-Whitney-Wilcoxon rank sum test is the non-parametric equivalent of the t-test.

6.3 Benchmark Functions

Various functions from the WFG toolkit are used to benchmark algorithm performance in this thesis. These functions ensure a diverse, unbiased testing environment due to the plethora of distinct shapes, biases and modalities incorporated by each. All functions selected also incorporate unique difficulties seen in practical MOPs, presenting a challenging task for MOOs. A comprehensive overview of the WFG toolkit can be found in Section 3.4.2. A summary of all WFG functions used in this work along with their properties is displayed in Table 6.1.

6.4 Algorithm Parameters

All algorithms used an identical number of candidate solutions and function evaluations to ensure unbiased comparisons as recommended in [34]. The number of candidate solutions used depends on the number of reference points employed for the

Table 6.1: Overview of WFG Functions Used

Function	Separability	Bias	Shape	Modality
1	Separable	Polynomial, Flat	Convex	Uni
2	Non-Separable	-	Convex, Disconnected	Uni, Multi
3	Non-Separable	-	Linear, Degenerate	Uni
4	Separable	-	Concave	Multi
5	Separable	-	Concave	Deceptive
6	Non-Separable	-	Concave	Uni
7	Separable	Parameter Dependent	Concave	Uni
8	Non-Separable	Parameter Dependent	Concave	Uni
9	Non-Separable	Parameter Dependent	Concave	Multi, Deceptive

decomposition-based algorithms. To generate uniformly spaced reference points for NSGA-III, MOEA/D and SrEA/D, we sample from a unit simplex using Das and Dennis's boundary intersection method [21]. Similar to [27], we employ a boundary layer and inside layer for problems with $n_c > 5$, since in those cases only using a boundary layer would result in an extremely large number of generated reference points. The number of generated reference points for varying n_c values are presented in Table 6.2, where D_1 and D_2 are the number of divisions in the boundary layer and inner layer, respectively. The number of candidate solutions employed for all algorithms is identical to the number of reference points generated.

The number of function evaluations used for each algorithm was dependent on the problem complexity and number of objectives, displayed in Table 6.3. All algorithms requiring non-dominated sorting employed the recently proposed efficient non-dominated sorting (ENS) [107] algorithm. Concerning the T parameter of the knee-driven algorithms, many values were tested for each experiment and the setting which produced the best IGD value on average over 10 runs was chosen. The exact T values used can be viewed in Table 6.4.

Table 6.2: Number of Reference Points Employed

n_c	D_1	D_2	# of reference points
2	99	0	100
4	7	0	120
6	4	1	132
8	3	2	156
10	3	2	275

Table 6.3: Number of Function Evaluations Used

Benchmark Function	$n_c = 2$	$n_c = 4$	$n_c = 6$	$n_c = 8$	$n_c = 10$
WFG1-2	20,000	35,000	45,000	50,000	65,000
WFG3-9	20,000	30,000	30,000	45,000	60,000

6.4.1 GA Parameters

All genetic algorithm variants within this work employ the simulated binary crossover [25] and polynomial mutation [26] as recombination operators with a distribution index of 20. Crossover probability was set to 1.0 and mutation probability was set to $1/n_x$. For selection, binary tournaments were employed in all cases. Regarding choice of aggregation function in MOEA/D, the PBI function was employed with $\theta = 5.0$ as recommended in [106]. The MOEA/D and SrEA/D algorithms utilized a neighbourhood range of $T = 20$.

6.4.2 PSO Parameters

SMPSO, CDAS-SMPSO and KnPSO each utilized a maximum archive size of 200 solutions. Regarding mutation, polynomial mutation [26] was used with a probability of $1/n_x$, identical to the evolutionary algorithms. In each iteration C_1 and C_2 varied randomly in the range [1.5,2.5] while ω varied randomly in the interval [0,0.8]. To handle boundary constraints, personal and neighbourhood bests were only updated if their positions remained within the legal bounds of the search space. Initial particle velocity was set to zero to comply with the recommendations given in [33].

Table 6.4: T values of KnPSO, KnEA and KnDE

	Number of Objectives				
Function	**2**	**4**	**6**	**8**	**10**
WFG4	0.6	0.5	0.5	0.3	0.2
WFG9	0.6	0.4	0.4	0.3	0.3
others	0.6	0.5	0.5	0.4	0.4

Table 6.5: S values of CDAS-SMPSO

	Number of Objectives				
Function	**2**	**4**	**6**	**8**	**10**
WFG1	0.35	0.30	0.35	0.35	0.30
WFG2	0.25	0.35	0.35	0.30	0.35
WFG3	0.25	0.45	0.40	0.40	0.30
WFG4	0.35	0.35	0.25	0.35	0.35
WFG5	0.45	0.40	0.40	0.40	0.40
WFG6	0.35	0.30	0.30	0.30	0.35
WFG7	0.40	0.30	0.35	0.35	0.30
WFG8	0.35	0.25	0.25	0.30	0.30
WFG9	0.30	0.25	0.35	0.35	0.40

Concerning the S value of CDAS-SMPSO, many values were tested for each experiment and the setting which produced the best IGD value on average was chosen. The exact S values used can be viewed in Table 6.5.

6.4.3 DE Parameters

GDE3 and KnDE each employed a CR and F value of 0.5.

Chapter 7

Knee Influence Experiments

This chapter discusses results obtained from the experiments which determine the influence of incorporating knee points. Section 7.1 consists of analyzing algorithm performance with regards to the hypervolume metric. In Section 7.2, results using the IGD metric are discussed.

7.1 Hypervolume

Tables 7.1-7.10 collectively present an overview of algorithmic performance with respect to the hypervolume metric over an increasing number of objectives. For the bi-objective test instances in Table 7.1, the knee-driven algorithms each obtained equal or worse perform than their non-knee counterparts. The KnPSO algorithm experienced notably poor performance, possessing the overall worst mean rank of 4.889, viewable in Table 7.2. However, when the number of objectives were increased past two, the inclusion of knee points were very beneficial in terms of increasing hypervolume performance. For the four objective instances in Table 7.4, the KnEA and KnPSO algorithms obtained mean ranks of 1.444 and 3.778 respectively, considerably better than their ranks on the two objective problems in Table 7.1. When the number of objectives were increased to six, given in Table 7.5, the knee-driven algorithms each obtained better performance than their non-knee counterparts, with KnEA obtaining

the overall best rank of 1.222 on average. The KnEA algorithm remains the best performing algorithm for the eight and ten objective test instances, given in Tables 7.7 and 7.9, respectively. In each of these instances, KnDE outperforms GDE3, KnPSO performs better than SMPSO and KnEA obtains better hypervolume values than NSGA-II.

From the observations above, it is clear that that the knee point approaches produced significantly higher hypervolume values in comparison to the non-knee point methods as the number of objectives increased. This observation is consistent with [108], which states that prioritizing knee points produces a higher hypervolume value. Note that the largest performance increase was seen in Tables 7.4, 7.6, 7.8 and 7.10, where the KnDE algorithm ranked approximately 2.5 positions better than the GDE3 algorithm on average over all test instances. From this, we conclude that DE seems to respond exceptionally well to the inclusion of knee points as an additional selection criterion in terms of hypervolume performance.

Figures 7.1-7.9 plot the rank of each algorithm over an increasing number of objective for problems WFG1-WFG9, respectively. For the WFG1 function, which possesses a flat bias and both convex and concave geometries, the KnEA algorithm consistently performed best overall. We note that the NSGA-II algorithm is the worst performing algorithm here, suggesting that WFG1 responded especially well to the knee point inclusion in KnEA. Another notable observation is present for the WFG3 function in Figure 7.3, where each knee-driven algorithm performed poorly. Here, it seems that the linear, degenerate landscape of WFG3 presented significant difficulties for the knee-driven optimizers, rendering KnEA, KnPSO and KnDE unable to produce satisfactory hypervolume values. WFG4-WFG9, given in Figures 7.4-7.9, seemed to possess the same general conclusion, namely that the knee point approaches scaled best by far as the number of objectives increased.

Overall, the inclusion of knee points improves hypervolume metric performance

as evidenced by the observations noted in this section. Superior hypervolume values indicate that the knee point identification mechanism used in KnEA, KnPSO and KnDE improve both diversity and Pareto-optimal front convergence to a degree. Functions with degenerate linear landscapes are the exception to this, as all knee point algorithms experienced significant performance degradation for the WFG3 problem.

Table 7.1: Mann-Whitney Wins and Losses For The Hypervolume Metric Using 30 Decision Variables, 2 Objectives

		WFG Function								
Algorithm	**Result**	1	2	3	4	5	6	7	8	9
KnDE	Wins	4	3	5	1	1	3	4	0	0
	Losses	0	0	0	1	0	1	0	3	2
	Difference	+4	+3	+5	0	+1	+2	+4	-3	-2
	Rank	1	1	1	3	1	2	1	6	3
GDE3	Wins	4	3	3	1	1	3	4	0	0
	Losses	0	0	1	2	0	1	0	2	2
	Difference	+4	+3	+2	-1	+1	+2	+4	-2	-2
	Rank	1	1	2	4	1	2	1	4	3
KnPSO	Wins	0	2	0	0	0	0	0	4	4
	Losses	5	3	5	5	5	5	5	1	1
	Difference	-5	-1	-5	-5	-5	-5	-5	+3	+3
	Rank	6	4	6	6	6	6	6	2	2
SMPSO	Wins	1	3	3	1	1	5	3	5	5
	Losses	4	0	1	2	0	0	2	0	0
	Difference	-3	+3	+2	-1	+1	+5	+1	+5	+5
	Rank	5	1	2	4	1	1	3	1	1
KnEA	Wins	2	0	1	4	1	1	1	0	0
	Losses	2	4	3	0	0	3	3	2	2
	Difference	0	-4	-2	+4	+1	-2	-2	-2	-2
	Rank	3	5	4	1	1	4	4	4	3
NSGA-II	Wins	2	0	1	3	1	1	1	1	0
	Losses	2	4	3	0	0	3	3	2	2
	Difference	0	-4	-2	+3	+1	-2	-2	-1	-2
	Rank	3	5	4	2	1	4	4	3	3

Table 7.2: Hypervolume Rank Summary Using 30 Decision Variables, 2 Objectives

	Algorithm					
Measure	KnDE	GDE3	KnPSO	SMPSO	KnEA	NSGA-II
Mean	**2.111**	**2.111**	4.889	**2.111**	3.222	3.222
Standard Deviation	1.595	1.197	1.663	1.449	1.315	1.133
Maximum	6.0	4.0	6.0	5.0	5.0	5.0
Minimum	1.0	1.0	2.0	1.0	1.0	1.0

Table 7.3: Mann-Whitney Wins and Losses For The Hypervolume Metric Using 30 Decision Variables, 4 Objectives

		WFG Function								
Algorithm	**Result**	1	2	3	4	5	6	7	8	9
KnDE	Wins	3	5	2	4	4	2	4	4	4
	Losses	1	0	3	1	1	3	1	1	1
	Difference	+2	+5	-1	+3	+3	-1	+3	+3	+3
	Rank	2	1	4	2	2	4	2	2	2
GDE3	Wins	1	3	0	1	1	0	0	0	1
	Losses	1	1	5	4	4	5	5	5	4
	Difference	0	+2	-5	-3	-3	-5	-5	-5	-3
	Rank	3	2	6	5	5	6	6	6	5
KnPSO	Wins	1	0	1	3	2	4	2	2	2
	Losses	2	5	4	2	3	1	2	3	2
	Difference	-1	-5	-3	+1	-1	+3	0	-1	0
	Rank	4	6	5	3	4	2	3	4	3
SMPSO	Wins	1	3	3	0	0	3	1	1	0
	Losses	2	1	2	5	5	2	4	4	5
	Difference	-1	+2	+1	-5	-5	+1	-3	-3	-5
	Rank	4	2	3	6	6	3	5	5	6
KnEA	Wins	5	2	4	5	5	5	5	5	5
	Losses	0	3	1	0	0	0	0	0	0
	Difference	+5	-1	+3	+5	+5	+5	+5	+5	+5
	Rank	1	4	2	1	1	1	1	1	1
NSGA-II	Wins	0	1	5	2	3	1	2	3	2
	Losses	5	4	0	3	2	4	2	2	2
	Difference	-5	-3	+5	-1	+1	-3	0	+1	0
	Rank	6	5	1	4	3	5	3	3	3

Table 7.4: Hypervolume Rank Summary Using 30 Decision Variables, 4 Objectives

	Algorithm					
Measure	KnDE	GDE3	KnPSO	SMPSO	KnEA	NSGA-II
Mean	2.333	4.889	3.778	4.444	**1.444**	3.667
Standard Deviation	0.943	1.37	1.133	1.423	0.956	1.414
Maximum	4.0	6.0	6.0	6.0	4.0	6.0
Minimum	1.0	2.0	2.0	2.0	1.0	1.0

Table 7.5: Mann-Whitney Wins and Losses For The Hypervolume Metric Using 30 Decision Variables, 6 Objectives

		WFG Function								
Algorithm	**Result**	1	2	3	4	5	6	7	8	9
KnDE	Wins	1	4	1	4	4	3	4	4	4
	Losses	1	0	4	1	1	2	1	1	1
	Difference	0	+4	-3	+3	+3	+1	+3	+3	+3
	Rank	2	1	5	2	2	3	2	2	2
GDE3	Wins	1	0	0	0	1	0	0	0	1
	Losses	1	4	5	4	4	5	5	5	4
	Difference	0	-4	-5	-4	-3	-5	-5	-5	-3
	Rank	2	6	6	5	5	6	6	6	5
KnPSO	Wins	1	0	2	3	2	4	3	2	3
	Losses	1	3	3	2	2	1	2	3	2
	Difference	0	-3	-1	+1	0	+3	+1	-1	+1
	Rank	2	5	4	3	3	2	3	4	3
SMPSO	Wins	1	2	3	0	0	2	1	1	0
	Losses	1	2	2	4	5	3	4	4	5
	Difference	0	0	+1	-4	-5	-1	-3	-3	-5
	Rank	2	3	3	5	6	4	5	5	6
KnEA	Wins	5	1	4	5	5	5	5	5	5
	Losses	0	1	0	0	0	0	0	0	0
	Difference	+5	0	+4	+5	+5	+5	+5	+5	+5
	Rank	1	3	1	1	1	1	1	1	1
NSGA-II	Wins	0	4	4	2	2	1	2	3	2
	Losses	5	1	0	3	2	4	3	2	3
	Difference	-5	+3	+4	-1	0	-3	-1	+1	-1
	Rank	6	2	1	4	3	5	4	3	4

Table 7.6: Hypervolume Rank Summary Using 30 Decision Variables, 6 Objectives

	Algorithm					
Measure	KnDE	GDE3	KnPSO	SMPSO	KnEA	NSGA-II
Mean	2.333	5.222	3.222	4.333	**1.222**	3.556
Standard Deviation	1.054	1.227	0.916	1.333	0.629	1.423
Maximum	5.0	6.0	5.0	6.0	3.0	6.0
Minimum	1.0	2.0	2.0	2.0	1.0	1.0

Table 7.7: Mann-Whitney Wins and Losses For The Hypervolume Metric Using 30 Decision Variables, 8 Objectives

		WFG Function								
Algorithm	**Result**	1	2	3	4	5	6	7	8	9
KnDE	Wins	1	3	0	4	4	3	4	4	4
	Losses	1	2	4	1	1	2	1	1	1
	Difference	0	+1	-4	+3	+3	+1	+3	+3	+3
	Rank	2	3	5	2	2	3	2	2	2
GDE3	Wins	1	0	0	1	1	0	0	0	0
	Losses	1	5	4	3	4	5	5	5	4
	Difference	0	-5	-4	-2	-3	-5	-5	-5	-4
	Rank	2	6	5	4	5	6	6	6	6
KnPSO	Wins	1	1	2	3	3	4	3	2	3
	Losses	1	4	3	2	2	1	2	2	2
	Difference	0	-3	-1	+1	+1	+3	+1	0	+1
	Rank	2	5	4	3	3	2	3	3	3
SMPSO	Wins	1	2	3	0	0	2	1	1	1
	Losses	1	3	2	4	5	3	3	4	3
	Difference	0	-1	+1	-4	-5	-1	-2	-3	-2
	Rank	2	4	3	6	6	4	4	5	4
KnEA	Wins	5	4	4	5	5	5	5	5	5
	Losses	0	1	1	0	0	0	0	0	0
	Difference	+5	+3	+3	+5	+5	+5	+5	+5	+5
	Rank	1	2	2	1	1	1	1	1	1
NSGA-II	Wins	0	5	5	0	2	1	1	2	0
	Losses	5	0	0	3	3	4	3	2	3
	Difference	-5	+5	+5	-3	-1	-3	-2	0	-3
	Rank	6	1	1	5	4	5	4	3	5

Table 7.8: Hypervolume Rank Summary Using 30 Decision Variables, 8 Objectives

	Algorithm					
Measure	KnDE	GDE3	KnPSO	SMPSO	KnEA	NSGA-II
Mean	2.556	5.111	3.111	4.222	**1.222**	3.778
Standard Deviation	0.956	1.286	0.875	1.227	0.416	1.685
Maximum	5.0	6.0	5.0	6.0	2.0	6.0
Minimum	2.0	2.0	2.0	2.0	1.0	1.0

Table 7.9: Mann-Whitney Wins and Losses For The Hypervolume Metric Using 30 Decision Variables, 10 Objectives

		WFG Function								
Algorithm	**Result**	1	2	3	4	5	6	7	8	9
KnDE	Wins	1	3	0	4	4	2	4	4	4
	Losses	1	2	4	1	1	2	1	1	1
	Difference	0	+1	-4	+3	+3	0	+3	+3	+3
	Rank	2	3	6	2	2	3	2	2	2
GDE3	Wins	1	0	1	1	1	0	0	0	0
	Losses	1	5	3	3	4	5	5	5	5
	Difference	0	-5	-2	-2	-3	-5	-5	-5	-5
	Rank	2	6	4	4	5	6	6	6	6
KnPSO	Wins	0	1	3	3	3	4	3	2	3
	Losses	1	4	2	2	2	1	2	2	2
	Difference	-1	-3	+1	+1	+1	+3	+1	0	+1
	Rank	4	5	3	3	3	2	3	3	3
SMPSO	Wins	0	2	4	0	0	2	1	1	2
	Losses	1	3	1	3	5	2	3	4	3
	Difference	-1	-1	+3	-3	-5	0	-2	-3	-1
	Rank	4	4	2	5	6	3	4	5	4
KnEA	Wins	5	5	0	5	5	5	5	5	5
	Losses	0	0	3	0	0	0	0	0	0
	Difference	+5	+5	-3	+5	+5	+5	+5	+5	+5
	Rank	1	1	5	1	1	1	1	1	1
NSGA-II	Wins	0	4	5	0	2	1	1	2	1
	Losses	3	1	0	4	3	4	3	2	4
	Difference	-3	+3	+5	-4	-1	-3	-2	0	-3
	Rank	6	2	1	6	4	5	4	3	5

Table 7.10: Hypervolume Rank Summary Using 30 Decision Variables, 10 Objectives

	Algorithm					
Measure	KnDE	GDE3	KnPSO	SMPSO	KnEA	NSGA-II
Mean	2.667	5.0	3.222	4.111	**1.444**	4.0
Standard Deviation	1.247	1.333	0.786	1.1	1.257	1.633
Maximum	6.0	6.0	5.0	6.0	5.0	6.0
Minimum	2.0	2.0	2.0	2.0	1.0	1.0

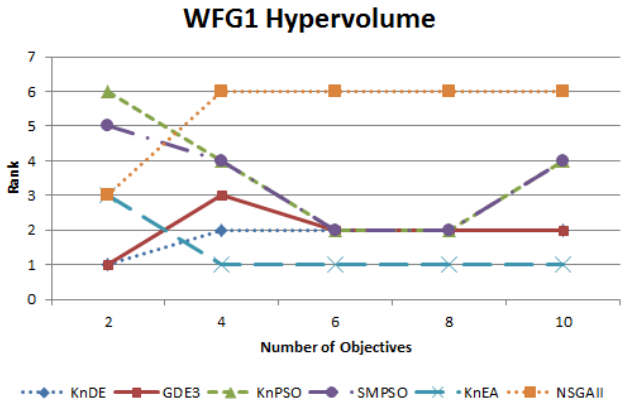

Figure 7.1: Ranking of hypervolume metric vs. number of objectives for the WFG1 problem

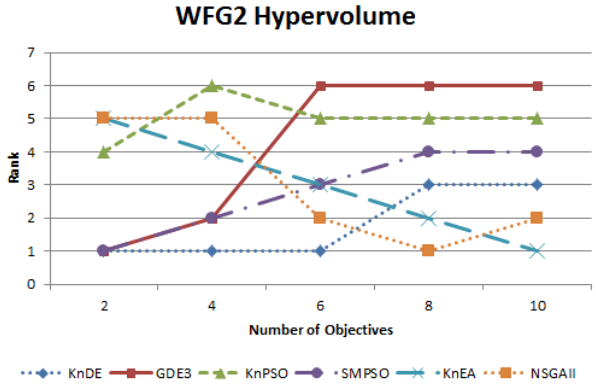

Figure 7.2: Ranking of hypervolume metric vs. number of objectives for the WFG2 problem

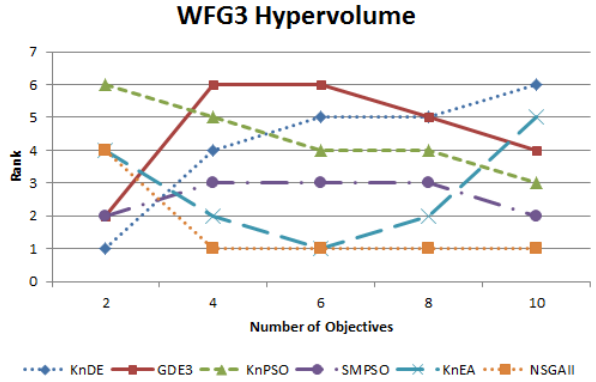

Figure 7.3: Ranking of hypervolume metric vs. number of objectives for the WFG3 problem

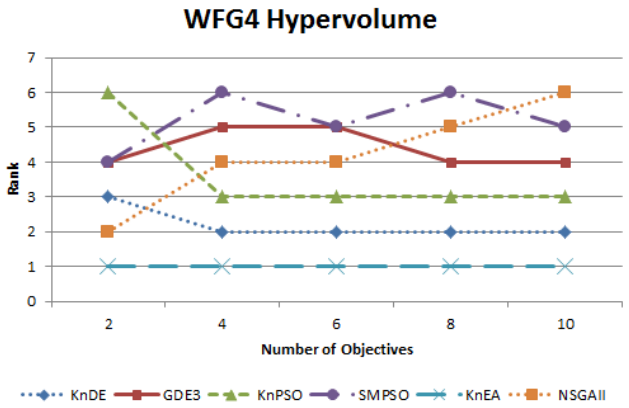

Figure 7.4: Ranking of hypervolume metric vs. number of objectives for the WFG4 problem

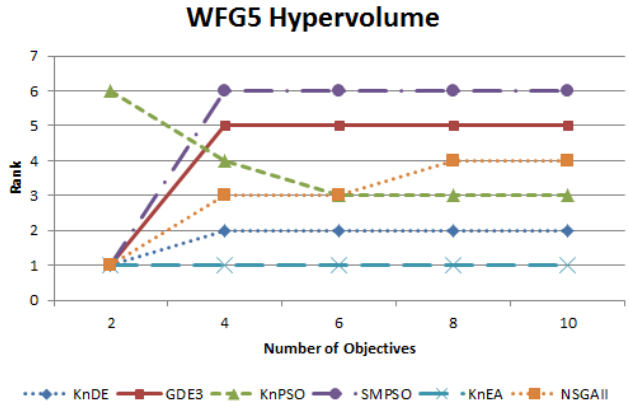

Figure 7.5: Ranking of hypervolume metric vs. number of objectives for the WFG5 problem

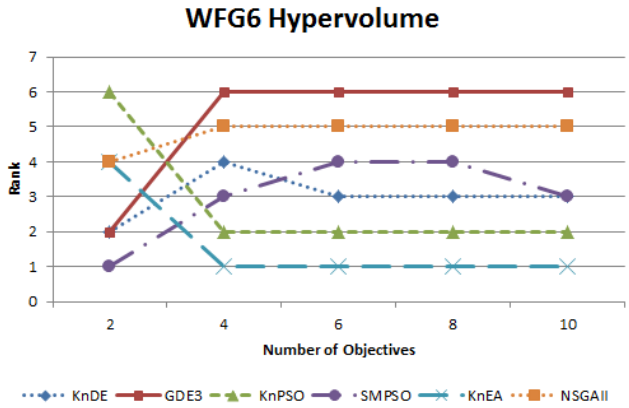

Figure 7.6: Ranking of hypervolume metric vs. number of objectives for the WFG6 problem

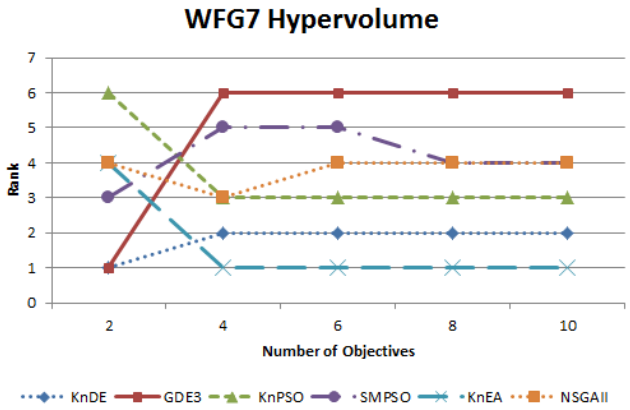

Figure 7.7: Ranking of hypervolume metric vs. number of objectives for the WFG7 problem

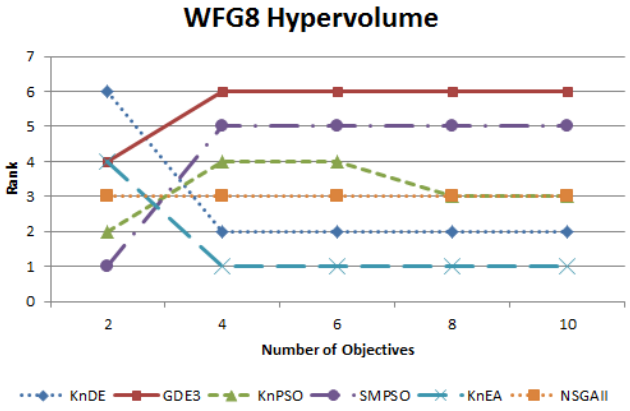

Figure 7.8: Ranking of hypervolume metric vs. number of objectives for the WFG8 problem

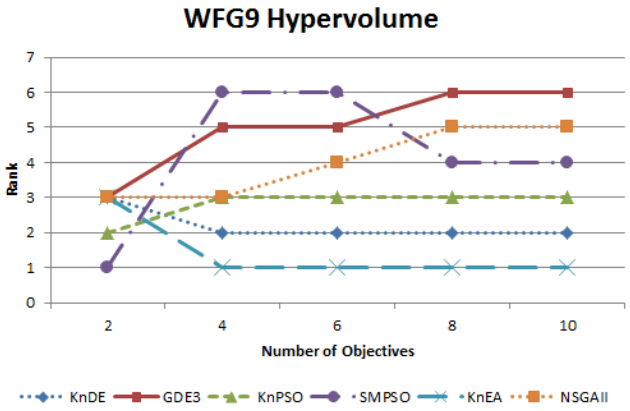

Figure 7.9: Ranking of hypervolume metric vs. number of objectives for the WFG9 problem

7.2 Inverted Generational Distance

Results of each algorithm over an increasing number of objectives with respect to the IGD metric is given in Tables 7.11-7.20. For the two, four and six objective test instances in Tables 7.11, 7.13 and 7.15, respectively, knee points seem to provide a significant performance boost for the PSO approach. In each of these tables, KnPSO obtains a better mean ranking than SMPSO, indicating that the inclusion of knee points were useful even for a low number of objectives. This observation was not the case for the evolutionary algorithms, as NSGA-II actually outperformed KnEA and KnDE performed marginally better than GDE3 in each case. When the number of objectives were increased past six, viewable in Figures 7.17-7.20, the knee point approaches each experienced much better rankings overall. For these eight objective test instances in Table 7.18, KnPSO yielded the best overall mean ranking of 2.111, while KnDE and KnEA both ranked approximately third overall on average. Regarding the ten objective test instance in Table 7.20, each knee-driven algorithm outperformed its non-knee counterpart, with the KnEA algorithm possessing the best mean rank of 2.444.

Examining Tables 7.11-7.20, one will observe that the overall performance of the KnEA algorithm is worse than its hypervolume performance in Section 7.1. KnEA obtained a higher mean rank than KnPSO for the six and eight objective test instances in Tables 7.16 and 7.18, respectively. Additionally, for the experiments where the number of objectives was set to ten (Table 7.20), KnEA only marginally outperforms KnPSO with a mean rank of 2.444. The performance disparity between the hypervolume and IGD values indicates that KnEA may cover a large region of the search space, however only a portion of the covered region is close to the true Pareto-optimal front.

The IGD ranks of each algorithm on WFG1-WFG9 are displayed in Figures 7.10-7.18, respectively. For the WFG1 funtion, displayed in Figure 7.10, the perfor-

mance of the KnEA algorithm is exceptionally good. When the number of objectives were six or greater, KnEA outperformed all other algorithms for WFG1. Regarding the WFG3 function (Figure 7.12), which is the connected version of WFG2, the GAs performed very well, with NSGA-II scaling best overall. The GAs also obtained the best IGD values for the non-separable WFG6 function and the parameter-dependent WFG7 problem, with the KnEA algorithm scaling best overall. Another notable observation was present for the WFG4 function, namely that the DE optimizers struggled to produce satisfactory IGD values as the number of objectives grew. Since WFG4 is multi-modal and possesses large "hills" which easily deceive MOOS, it is likely that both KnDE and GDE3 became trapped in a local minima at an early stage of the search. For the deceptive WFG5 problem and the non-separable WFG9 problem in Figures 7.14 and 7.18, respectively, the PSO algorithms were the highest performing approaches. The KnPSO algorithm produced the best overall IGD metric values for each of these functions.

In summary, results of the IGD experiments performed suggest that the utilization of knee points is a formidable solution for improving the performance of Pareto optimizers on MaOPs. Incorporating knee points during optimization aided in producing higher IGD metric values, indicative of better selection pressure towards the Pareto-optimal front. Thus, each knee-driven algorithm was able to obtain a better spread of solutions and convergence to the true front.

Table 7.11: Mann-Whitney Wins and Losses For The IGD Metric Using 30 Decision Variables, 2 Objectives

		WFG Function								
Algorithm	**Result**	1	2	3	4	5	6	7	8	9
KnDE	Wins	4	2	5	2	3	3	3	3	0
	Losses	0	0	0	2	2	1	0	1	5
	Difference	+4	+2	+5	0	+1	+2	+3	+2	-5
	Rank	1	1	1	3	3	2	1	2	6
GDE3	Wins	4	2	3	2	4	3	2	3	1
	Losses	0	0	2	2	1	1	3	1	4
	Difference	+4	+2	+1	0	+3	+2	-1	+2	-3
	Rank	1	1	3	3	2	2	4	2	5
KnPSO	Wins	3	1	1	1	5	5	1	0	5
	Losses	2	4	3	4	0	0	4	3	0
	Difference	+1	-3	-2	-3	+5	+5	-3	-3	+5
	Rank	3	5	4	5	1	1	5	5	1
SMPSO	Wins	2	0	0	0	0	0	0	5	4
	Losses	3	5	5	5	5	5	5	0	1
	Difference	-1	-5	-5	-5	-5	-5	-5	+5	+3
	Rank	4	6	6	6	6	6	6	1	2
KnEA	Wins	0	2	4	4	2	1	3	0	2
	Losses	5	0	1	0	3	3	0	4	2
	Difference	-5	+2	+3	+4	-1	-2	+3	-4	0
	Rank	6	1	2	1	4	4	1	6	3
NSGA-II	Wins	1	2	1	4	1	1	3	1	2
	Losses	4	0	3	0	4	3	0	3	2
	Difference	-3	+2	-2	+4	-3	-2	+3	-2	0
	Rank	5	1	4	1	5	4	1	4	3

Table 7.12: IGD Rank Summary Using 30 Decision Variables, 2 Objectives

	Algorithm					
Measure	KnDE	GDE3	KnPSO	SMPSO	KnEA	NSGA-II
Mean	**2.222**	2.556	3.333	4.778	3.111	3.111
Standard Deviation	1.548	1.257	1.872	1.764	1.912	1.595
Maximum	6.0	5.0	6.0	5.0	6.0	5.0
Minimum	1.0	1.0	1.0	1.0	1.0	1.0

Table 7.13: Mann-Whitney Wins and Losses For The IGD Metric Using 30 Decision Variables, 4 Objectives

		WFG Function								
Algorithm	**Result**	1	2	3	4	5	6	7	8	9
KnDE	Wins	5	4	1	2	4	3	4	2	3
	Losses	0	1	3	2	0	2	0	2	1
	Difference	+5	+3	-2	0	+4	+1	+4	0	+2
	Rank	1	2	4	3	1	3	1	3	2
GDE3	Wins	4	0	0	1	3	0	1	2	3
	Losses	1	5	5	3	2	5	4	2	1
	Difference	+3	-5	-5	-2	+1	-5	-3	0	+2
	Rank	2	6	6	5	3	6	5	3	2
KnPSO	Wins	3	3	3	4	0	2	2	4	1
	Losses	2	2	2	0	4	3	2	0	3
	Difference	+1	+1	+1	+4	-4	-1	0	+4	-2
	Rank	3	3	3	1	5	4	3	1	4
SMPSO	Wins	2	1	1	0	0	1	0	0	0
	Losses	3	4	3	5	4	4	5	5	5
	Difference	-1	-3	-2	-5	-4	-3	-5	-5	-5
	Rank	4	5	4	6	5	5	6	6	6
KnEA	Wins	1	5	5	1	2	4	2	1	1
	Losses	4	0	0	2	3	0	2	4	3
	Difference	-3	+5	+5	-1	-1	+4	0	-3	-2
	Rank	5	1	1	4	4	1	3	5	4
NSGA-II	Wins	0	2	4	4	4	4	4	4	5
	Losses	5	3	1	0	0	0	0	0	0
	Difference	-5	-1	+3	+4	+4	+4	+4	+4	+5
	Rank	6	4	2	1	1	1	1	1	1

Table 7.14: IGD Rank Summary Using 30 Decision Variables, 4 Objectives

	Algorithm					
Measure	KnDE	GDE3	KnPSO	SMPSO	KnEA	NSGA-II
Mean	2.222	4.222	3.0	5.222	3.111	**2.0**
Standard Deviation	1.03	1.618	0.786	1.247	1.595	1.7
Maximum	4.0	6.0	6.0	5.0	5.0	6.0
Minimum	1.0	2.0	4.0	1.0	1.0	1.0

Table 7.15: Mann-Whitney Wins and Losses For The IGD Metric Using 30 Decision Variables, 6 Objectives

		WFG Function								
Algorithm	**Result**	1	2	3	4	5	6	7	8	9
KnDE	Wins	3	3	0	1	2	3	2	2	5
	Losses	1	1	5	2	2	2	3	3	0
	Difference	+2	+2	-5	-1	0	+1	-1	-1	+5
	Rank	2	2	6	3	4	3	4	4	1
GDE3	Wins	3	0	1	0	3	0	3	4	0
	Losses	1	4	4	3	0	3	2	1	5
	Difference	+2	-4	-3	-3	+3	-3	+1	+3	-5
	Rank	2	5	5	6	2	4	3	2	6
KnPSO	Wins	2	2	3	5	4	5	4	5	1
	Losses	3	3	2	0	0	0	0	0	3
	Difference	-1	-1	+1	+5	+4	+5	+4	+5	-2
	Rank	4	4	3	1	1	1	1	1	5
SMPSO	Wins	1	0	2	0	1	0	1	1	1
	Losses	4	4	3	2	4	3	4	4	1
	Difference	-3	-4	-1	-2	-3	-3	-3	-3	0
	Rank	5	5	4	4	5	4	5	5	4
KnEA	Wins	5	5	5	0	0	0	0	0	2
	Losses	0	0	0	2	5	3	5	5	1
	Difference	+5	+5	+5	-2	-5	-3	-5	-5	+1
	Rank	1	1	1	4	6	4	6	6	2
NSGA-II	Wins	0	3	4	4	2	4	4	3	2
	Losses	5	1	1	1	1	1	0	2	1
	Difference	-5	+2	+3	+3	+1	+3	+4	+1	+1
	Rank	6	2	2	2	3	2	1	3	2

Table 7.16: IGD Rank Summary Using 30 Decision Variables, 6 Objectives

	Algorithm					
Measure	KnDE	GDE3	KnPSO	SMPSO	KnEA	NSGA-II
Mean	3.222	3.889	**2.333**	4.556	3.444	2.556
Standard Deviation	1.397	1.595	0.497	1.563	2.114	1.343
Maximum	6.0	6.0	5.0	5.0	6.0	6.0
Minimum	1.0	2.0	4.0	1.0	1.0	1.0

Table 7.17: Mann-Whitney Wins and Losses For The IGD Metric Using 30 Decision Variables, 8 Objectives

		WFG Function								
Algorithm	**Result**	1	2	3	4	5	6	7	8	9
KnDE	Wins	3	3	0	1	2	3	2	1	5
	Losses	1	2	5	4	0	2	2	3	0
	Difference	+2	+1	-5	-3	+2	+1	0	-2	+5
	Rank	2	3	6	5	1	3	3	4	1
GDE3	Wins	3	0	1	0	2	1	1	4	0
	Losses	1	5	4	5	0	4	3	1	4
	Difference	+2	-5	-3	-5	+2	-3	-2	+3	-4
	Rank	2	6	5	6	1	5	5	2	6
KnPSO	Wins	2	2	3	3	2	5	5	5	1
	Losses	3	3	2	0	0	0	0	0	2
	Difference	-1	-1	+1	+3	+2	+5	+5	+5	-1
	Rank	4	4	3	1	1	1	1	1	3
SMPSO	Wins	1	1	2	3	1	2	1	1	0
	Losses	4	4	3	0	4	3	2	3	2
	Difference	-3	-3	-1	+3	-3	-1	-1	-2	-2
	Rank	5	5	4	1	5	4	4	4	5
KnEA	Wins	5	5	4	3	0	0	0	0	4
	Losses	0	0	0	0	5	5	5	5	1
	Difference	+5	+5	+4	+3	-5	-5	-5	-5	+3
	Rank	1	1	1	1	6	6	6	6	2
NSGA-II	Wins	0	4	4	2	2	4	4	3	1
	Losses	5	1	0	3	0	1	1	2	2
	Difference	-5	+3	+4	-1	+2	+3	+3	+1	-1
	Rank	6	2	1	4	1	2	2	3	3

Table 7.18: IGD Rank Summary Using 30 Decision Variables, 8 Objectives

	Algorithm					
Measure	KnDE	GDE3	KnPSO	SMPSO	KnEA	NSGA-II
Mean	3.111	4.222	**2.111**	4.111	3.333	2.667
Standard Deviation	1.595	1.872	1.197	1.286	2.404	1.491
Maximum	6.0	6.0	5.0	4.0	6.0	6.0
Minimum	1.0	1.0	1.0	1.0	1.0	1.0

Table 7.19: Mann-Whitney Wins and Losses For The IGD Metric Using 30 Decision Variables, 10 Objectives

		WFG Function								
Algorithm	**Result**	1	2	3	4	5	6	7	8	9
KnDE	Wins	3	3	0	1	0	1	1	4	0
	Losses	1	0	5	4	3	4	3	0	5
	Difference	+2	+3	-5	-3	-3	-3	-2	+4	-5
	Rank	2	1	6	5	4	5	4	1	6
GDE3	Wins	3	1	1	0	0	0	0	4	1
	Losses	1	3	4	5	3	5	5	0	2
	Difference	+2	-2	-3	-5	-3	-5	-5	+4	-1
	Rank	2	4	5	6	4	6	6	1	5
KnPSO	Wins	2	2	3	4	5	2	3	2	4
	Losses	3	2	2	0	0	2	2	3	0
	Difference	-1	0	+1	+4	+5	0	+1	-1	+4
	Rank	4	3	3	1	1	3	3	4	1
SMPSO	Wins	1	1	2	2	4	2	1	3	2
	Losses	4	3	3	2	1	2	3	2	1
	Difference	-3	-2	-1	0	+3	0	-2	+1	+1
	Rank	5	4	4	3	2	3	4	3	2
NSGA-II	Wins	0	3	5	2	0	4	4	0	1
	Losses	5	0	0	2	3	1	1	4	0
	Difference	-5	+3	+5	0	-3	+3	+3	-4	+1
	Rank	6	1	1	3	4	2	2	5	2
KnEA	Wins	5	0	4	4	3	5	5	0	1
	Losses	0	2	1	0	2	0	0	4	1
	Difference	+5	-2	+3	+4	+1	+5	+5	-4	0
	Rank	1	4	2	1	3	1	1	5	4

Table 7.20: IGD Rank Summary Using 30 Decision Variables, 10 Objectives

	Algorithm					
Measure	KnDE	GDE3	KnPSO	SMPSO	KnEA	NSGA-II
Mean	3.778	4.333	2.556	3.333	**2.444**	2.889
Standard Deviation	1.872	1.7	1.165	0.943	1.663	1.499
Maximum	6.0	6.0	4.0	5.0	6.0	5.0
Minimum	1.0	1.0	1.0	2.0	1.0	1.0

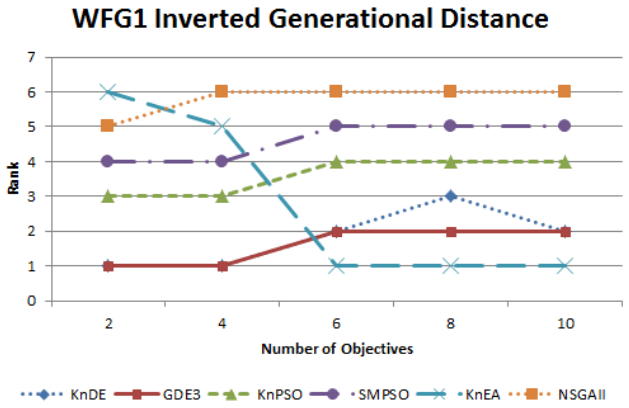

Figure 7.10: Ranking of IGD metric vs. number of objectives for the WFG1 problem

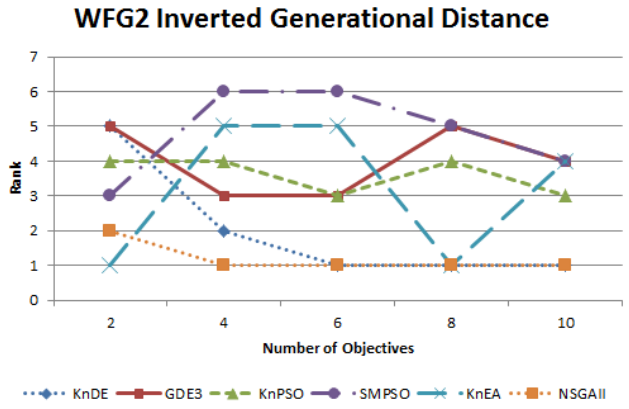

Figure 7.11: Ranking of IGD metric vs. number of objectives for the WFG2 problem

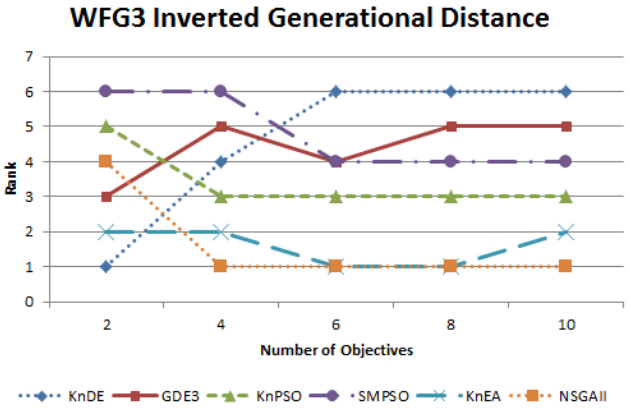

Figure 7.12: Ranking of IGD metric vs. number of objectives for the WFG3 problem

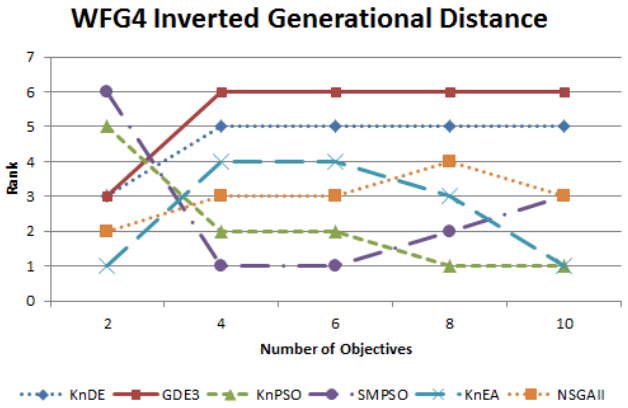

Figure 7.13: Ranking of IGD metric vs. number of objectives for the WFG4 problem

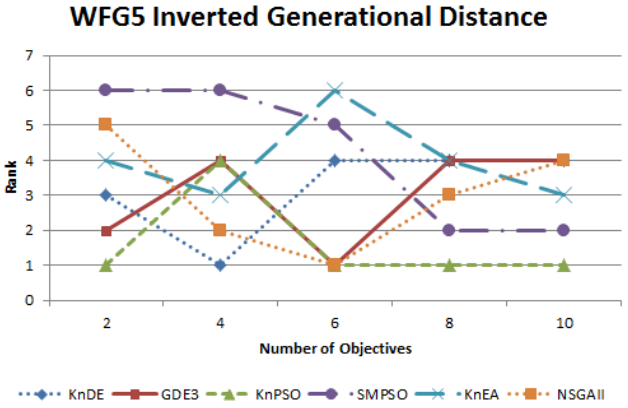

Figure 7.14: Ranking of IGD metric vs. number of objectives for the WFG5 problem

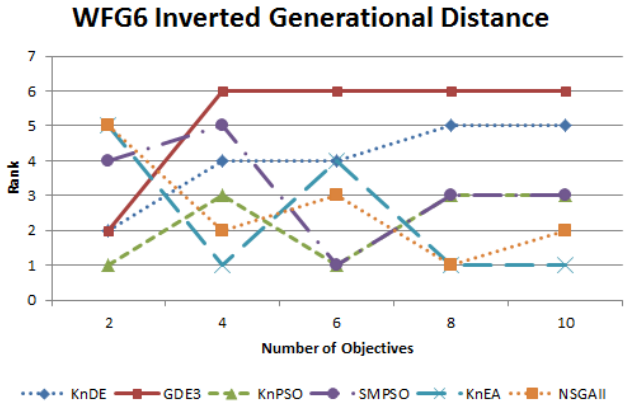

Figure 7.15: Ranking of IGD metric vs. number of objectives for the WFG6 problem

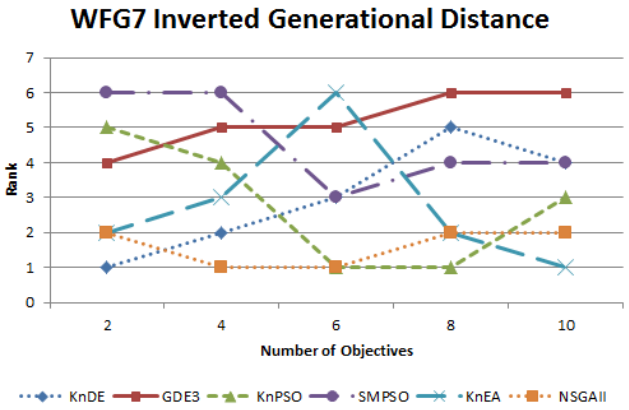

Figure 7.16: Ranking of IGD metric vs. number of objectives for the WFG7 problem

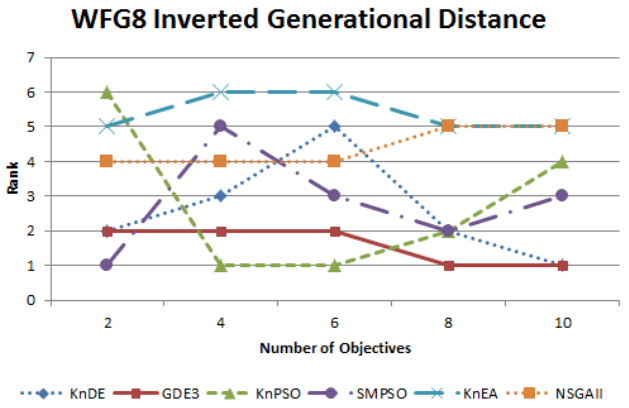

Figure 7.17: Ranking of IGD metric vs. number of objectives for the WFG8 problem

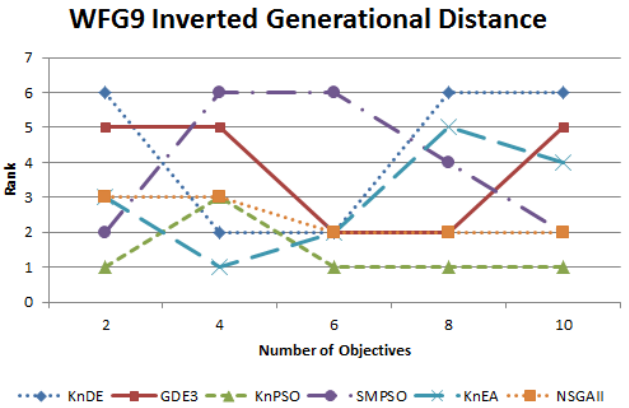

Figure 7.18: Ranking of IGD metric vs. number of objectives for the WFG9 problem

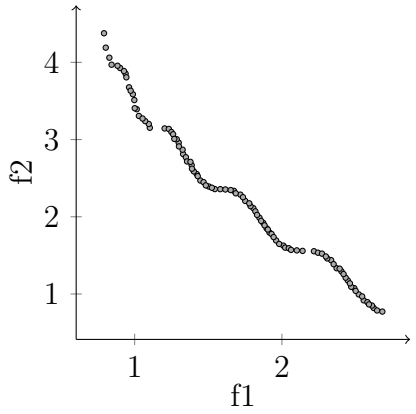

Figure 7.19: Sample Pareto front produced by GDE3 on WFG1.

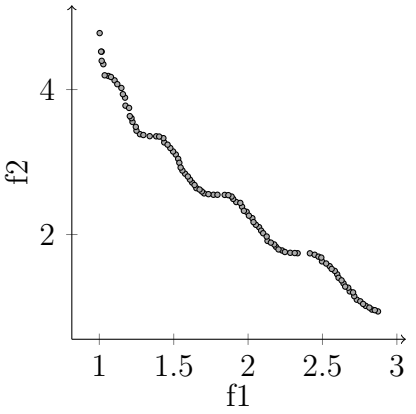

Figure 7.20: Sample Pareto front produced by SMPSO on WFG1.

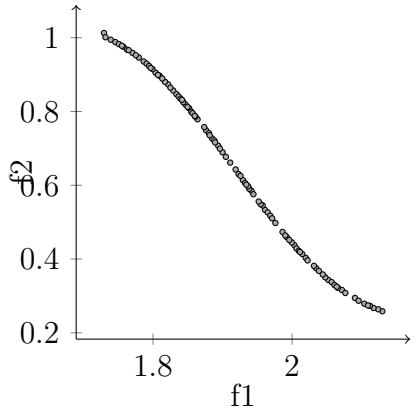

Figure 7.21: Sample Pareto front produced by NSGA-II on WFG1.

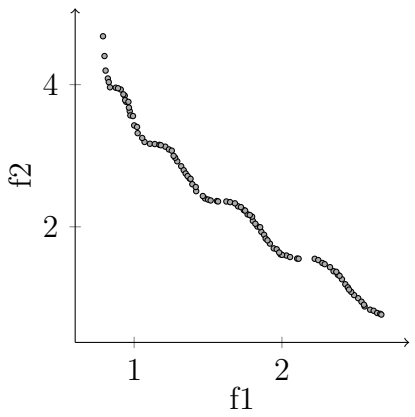

Figure 7.22: Sample Pareto front produced by KnDE on WFG1.

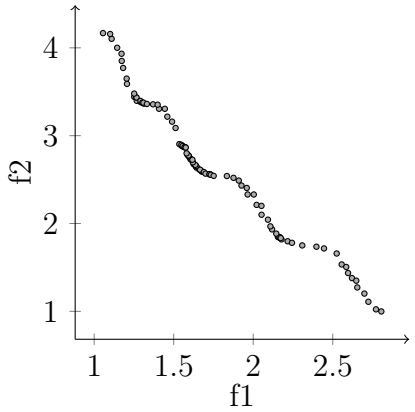

Figure 7.23: Sample Pareto front produced by KnPSO on WFG1.

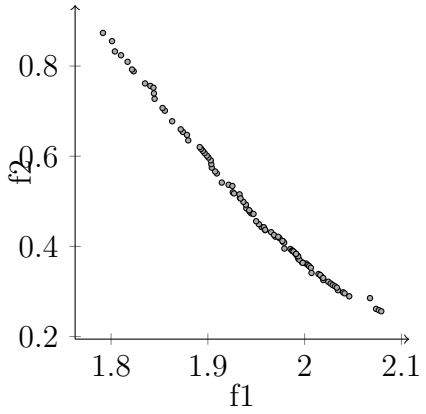

Figure 7.24: Sample Pareto front produced by KnEA on WFG1.

Chapter 8

Objective Scalability Experiments

This chapter presents a set of experiments analyzing the scalability of each algorithm as the number of objectives grow. Section 8.1 consists of analyzing algorithm performance with regards to the hypervolume metric. In Section 8.2, the IGD metric performance of each algorithm is discussed.

8.1 Hypervolume

Results of hypervolume metric performance over an increasing number of objectives is presented for each algorithm in Tables 8.1-8.10. Regarding the bi-objective problems in Table 8.2, the MOEA/D algorithm performed best with a mean rank of 2.222. Here, both KnDE and KnPSO performed notably bad with mean ranks of 5.778 and 4.556, respectively. One should note that the minimum rank of both algorithms were greater than 1.0, indicating that neither was the best performing algorithm for any of the WFG problems. For the test instances where the number of objectives were greater than two, i.e., Tables 8.3-8.10, the KnEA algorithm obtained the best mean hypervolume rank in all cases. KnEA frequently outperformed all other algorithms, winning 198/216 total comparisons for these tables. The decomposition-based MOEA/D and SrEA/D algorithms performed second and third best for all instances in Tables 8.3-8.10, obtaining slightly worse hypervolume ranks in comparison to KnEA.

It is quite clear that KnEA, MOEA/D and SrEA/D scale significantly better in comparison to all other algorithms with respect to hypervolume performance. A notable observation is the low standard deviation of the ranks seen for the MOEA/D algorithm, especially for the ten objective case seen in Table 8.10 where the standard deviation was 0.685. This observation indicates that the MOEA/D algorithm performed quite consistently, responding well to the unique difficulties of each WFG problem presented.

The overall worst performing algorithm is CDAS-SMPSO, as it obtained the highest hypervolume ranks on average for the four, eight and ten objective instances, viewable in Tables 8.4, 8.8 and 8.10, respectively. The KnDE algorithm performed notably bad when the number of objectives were low, experiencing its lowest mean values of 5.778, 6.0 and 5.0 for the two, four and six objective cases. When the number of objectives were increased past six, KnPSO experienced worse mean hypervolume ranks than KnDE, possessing an average of 4.667 and 4.889 for the eight and ten objective instances in Tables 8.8 and 8.10, respectively.

Figures 8.1-8.9 present a visualization of algorithmic performance on each WFG function as the number of objectives grow. Analyzing these figures, one will observe that KnEA was the best performing algorithm for all problems except the WFG3 function (Figure 8.3), which possesses a linear and degenerate landscape. Both KnDE and KnPSO also experienced difficulties on WFG3, supporting the conclusion made in Section 7.1 which stated that the hypervolume performance of knee approaches degrade in the presence of the degenerate linear function landscapes. We also note that KnPSO exhibited notably poor performance on WFG2, WFG3 and WFG5 as evidenced by Figures 8.2, 8.3 and 8.5 respectively. Concerning the decomposition-based approaches, which are NSGA-III, SrEA/D and MOEA/D, several WFG functions in particular seemed to respond very well to decomposition, namely WFG2, WFG7 and WFG9. NSGA-III and MOEA/D were especially effective for the non-separable

WFG2 function, whose Pareto-optimal front consists of several disconnected convex segments.

In conclusion, it is quite clear that the KnEA algorithm performs best in terms of the hypervolume metric as the number of objectives increase. The next best algorithms were MOEA/D and SrEA/D, who each experienced marginally worse mean rankings than KnEA in most instances. We note that the decomposition approach of MOEA/D seemed to perform quite consistently for all functions, leading one to conclude that its hypervolume performance was not degraded by any particular function landscape. Performance of CDAS-SMPSO, KnDE and KnPSO was subpar in comparison to the other algorithms.

Table 8.1: Mann-Whitney Wins and Losses For The Hypervolume Metric Using 30 Decision Variables, 2 Objectives

		WFG Function								
Algorithm	**Result**	1	2	3	4	5	6	7	8	9
NSGA-III	Wins	0	0	3	3	2	1	1	1	0
	Losses	6	3	3	2	0	3	4	3	3
	Difference	-6	-3	0	+1	+2	-2	-3	-2	-3
	Rank	7	4	4	3	2	5	6	4	5
MOEA/D	Wins	4	0	4	5	4	3	5	4	1
	Losses	1	3	1	0	0	1	1	1	2
	Difference	+3	-3	+3	+5	+4	+2	+4	+3	-1
	Rank	2	4	2	1	1	2	2	2	4
KnDE	Wins	1	0	0	2	1	0	0	0	3
	Losses	5	3	6	4	3	5	6	6	2
	Difference	-4	-3	-6	-2	-2	-5	-6	-6	+1
	Rank	6	4	7	5	6	7	7	7	3
KnPSO	Wins	2	4	1	0	0	0	1	4	5
	Losses	3	2	5	5	6	4	3	1	1
	Difference	-1	+2	-4	-5	-6	-4	-2	+3	+4
	Rank	4	3	6	6	7	6	5	2	2
SrEA/D	Wins	6	6	6	3	1	6	6	1	0
	Losses	0	0	0	2	1	0	0	3	4
	Difference	+6	+6	+6	+1	0	+6	+6	-2	-4
	Rank	1	1	1	3	4	1	1	4	7
CDAS-SMPSO	Wins	2	5	2	0	1	2	2	6	6
	Losses	3	1	4	5	1	1	2	0	0
	Difference	-1	+4	-2	-5	0	+1	0	+6	+6
	Rank	4	2	5	6	4	4	4	1	1
KnEA	Wins	4	0	4	5	2	3	3	1	0
	Losses	1	3	1	0	0	1	2	3	3
	Difference	+3	-3	+3	+5	+2	+2	+1	-2	-3
	Rank	2	4	2	1	2	2	3	4	5

Table 8.2: Hypervolume Rank Summary Using 30 Decision Variables, 2 Objectives

	Algorithm						
Measure	NSGA-III	MOEA/D	KnDE	KnPSO	SrEA/D	CDAS-SMPSO	KnEA
Mean	4.444	**2.222**	5.778	4.556	2.556	3.444	2.778
Standard Deviation	1.423	1.03	1.397	1.771	2.006	1.641	1.227
Maximum	7.0	4.0	7.0	7.0	7.0	6.0	5.0
Minimum	2.0	1.0	3.0	2.0	1.0	1.0	1.0

Table 8.3: Mann-Whitney Wins and Losses For The Hypervolume Metric Using 30 Decision Variables, 4 Objectives

		WFG Function								
Algorithm	**Result**	1	2	3	4	5	6	7	8	9
NSGA-III	Wins	0	2	3	2	4	3	3	1	4
	Losses	6	2	2	3	2	2	3	3	1
	Difference	-6	0	+1	-1	+2	+1	0	-2	+3
	Rank	7	3	3	4	3	3	4	5	2
MOEA/D	Wins	5	1	6	4	5	1	5	5	3
	Losses	1	3	0	1	1	4	1	0	3
	Difference	+4	-2	+6	+3	+4	-3	+4	+5	0
	Rank	2	6	1	2	2	5	2	1	4
KnDE	Wins	1	0	1	0	2	0	1	0	0
	Losses	5	6	4	6	4	6	5	6	4
	Difference	-4	-6	-3	-6	-2	-6	-4	-6	-4
	Rank	6	7	5	7	5	7	6	7	6
KnPSO	Wins	3	2	0	2	1	5	2	2	1
	Losses	3	2	6	3	5	1	4	3	4
	Difference	0	0	-6	-1	-4	+4	-2	-1	-3
	Rank	4	3	7	4	6	2	5	4	5
SrEA/D	Wins	4	6	3	4	3	1	4	4	4
	Losses	2	0	2	1	3	4	2	2	1
	Difference	+2	+6	+1	+3	0	-3	+2	+2	+3
	Rank	3	1	3	2	4	5	3	3	2
CDAS-SMPSO	Wins	2	1	1	1	0	3	0	1	0
	Losses	4	2	4	5	6	2	6	4	5
	Difference	-2	-1	-3	-4	-6	+1	-6	-3	-5
	Rank	5	5	5	6	7	3	7	6	7
KnEA	Wins	6	4	5	6	6	6	6	5	6
	Losses	0	1	1	0	0	0	0	0	0
	Difference	+6	+3	+4	+6	+6	+6	+6	+5	+6
	Rank	1	2	2	1	1	1	1	1	1

Table 8.4: Hypervolume Rank Summary Using 30 Decision Variables, 4 Objectives

	Algorithm						
Measure	NSGA-III	MOEA/D	KnDE	KnPSO	SrEA/D	CDAS-SMPSO	KnEA
Mean	3.778	2.778	6.0	4.444	2.889	5.667	**1.222**
Standard Deviation	1.397	1.685	1.247	1.423	1.286	1.247	0.416
Maximum	7.0	6.0	7.0	7.0	5.0	7.0	2.0
Minimum	2.0	1.0	3.0	2.0	1.0	3.0	1.0

Table 8.5: Mann-Whitney Wins and Losses For The Hypervolume Metric Using 30 Decision Variables, 6 Objectives

		WFG Function								
Algorithm	**Result**	1	2	3	4	5	6	7	8	9
NSGA-III	Wins	0	0	0	2	4	3	1	4	3
	Losses	6	5	5	4	2	2	3	2	2
	Difference	-6	-5	-5	-2	+2	+1	-2	+2	+1
	Rank	7	6	6	5	3	3	4	3	3
MOEA/D	Wins	5	2	5	5	5	3	5	5	3
	Losses	1	0	0	1	1	2	1	1	2
	Difference	+4	+2	+5	+4	+4	+1	+4	+4	+1
	Rank	2	3	1	2	2	3	2	2	3
KnDE	Wins	1	2	0	0	2	0	1	1	1
	Losses	5	2	5	5	4	6	3	5	4
	Difference	-4	0	-5	-5	-2	-6	-2	-4	-3
	Rank	6	5	6	6	5	7	4	6	5
KnPSO	Wins	2	0	2	3	1	5	1	2	1
	Losses	2	5	4	3	5	1	3	4	4
	Difference	0	-5	-2	0	-4	+4	-2	-2	-3
	Rank	3	6	5	4	6	2	4	5	5
SrEA/D	Wins	2	4	5	4	3	1	4	3	5
	Losses	2	0	0	2	3	4	2	3	1
	Difference	0	+4	+5	+2	0	-3	+2	0	+4
	Rank	3	1	1	3	4	5	3	4	2
CDAS-SMPSO	Wins	2	2	3	0	0	1	0	0	0
	Losses	2	1	3	5	6	4	6	6	6
	Difference	0	+1	0	-5	-6	-3	-6	-6	-6
	Rank	3	4	4	6	7	5	7	7	7
KnEA	Wins	6	3	4	6	6	6	6	6	6
	Losses	0	0	2	0	0	0	0	0	0
	Difference	+6	+3	+2	+6	+6	+6	+6	+6	+6
	Rank	1	2	3	1	1	1	1	1	1

Table 8.6: Hypervolume Rank Summary Using 30 Decision Variables, 6 Objectives

	Algorithm						
Measure	NSGA-III	MOEA/D	KnDE	KnPSO	SrEA/D	CDAS-SMPSO	KnEA
Mean	4.444	2.222	5.556	4.444	2.889	5.556	**1.333**
Standard Deviation	1.499	0.629	1.633	1.257	1.423	1.499	0.667
Maximum	7.0	3.0	7.0	6.0	6.0	7.0	3.0
Minimum	3.0	1.0	1.0	2.0	1.0	3.0	1.0

Table 8.7: Mann-Whitney Wins and Losses For The Hypervolume Metric Using 30 Decision Variables, 8 Objectives

		WFG Function								
Algorithm	**Result**	1	2	3	4	5	6	7	8	9
NSGA-III	Wins	0	0	3	2	2	1	1	4	3
	Losses	6	5	3	4	3	3	3	1	3
	Difference	-6	-5	0	-2	-1	-2	-2	+3	0
	Rank	7	6	4	5	4	4	4	2	4
MOEA/D	Wins	4	3	4	5	5	1	4	4	4
	Losses	1	1	1	1	1	3	1	1	1
	Difference	+3	+2	+3	+4	+4	-2	+3	+3	+3
	Rank	2	2	2	2	2	4	2	2	2
KnDE	Wins	4	3	0	0	2	0	1	1	2
	Losses	1	1	6	5	3	6	3	5	4
	Difference	+3	+2	-6	-5	-1	-6	-2	-4	-2
	Rank	2	2	7	6	4	7	4	6	5
KnPSO	Wins	1	0	1	3	1	5	1	2	1
	Losses	3	5	4	3	5	1	3	4	5
	Difference	-2	-5	-3	0	-4	+4	-2	-2	-4
	Rank	4	6	5	4	6	2	4	5	6
SrEA/D	Wins	1	3	6	4	4	1	4	3	4
	Losses	3	1	0	2	2	3	1	3	1
	Difference	-2	+2	+6	+2	+2	-2	+3	0	+3
	Rank	4	2	1	3	3	4	2	4	2
CDAS-SMPSO	Wins	1	2	1	0	0	4	0	0	0
	Losses	3	4	4	5	6	2	6	6	6
	Difference	-2	-2	-3	-5	-6	+2	-6	-6	-6
	Rank	4	5	5	6	7	3	7	7	7
KnEA	Wins	6	6	4	6	6	6	6	6	6
	Losses	0	0	1	0	0	0	0	0	0
	Difference	+6	+6	+3	+6	+6	+6	+6	+6	+6
	Rank	1	1	2	1	1	1	1	1	1

Table 8.8: Hypervolume Rank Summary Using 30 Decision Variables, 8 Objectives

	Algorithm						
Measure	NSGA-III	MOEA/D	KnDE	KnPSO	SrEA/D	CDAS-SMPSO	KnEA
Mean	4.444	2.222	4.778	4.667	2.778	5.667	**1.111**
Standard Deviation	1.343	0.629	1.969	1.247	1.499	1.414	0.314
Maximum	7.0	4.0	7.0	6.0	7.0	7.0	2.0
Minimum	2.0	2.0	1.0	2.0	2.0	3.0	1.0

Table 8.9: Mann-Whitney Wins and Losses For The Hypervolume Metric Using 30 Decision Variables, 10 Objectives

		WFG Function								
Algorithm	**Result**	1	2	3	4	5	6	7	8	9
NSGA-III	Wins	2	4	5	1	3	1	3	4	5
	Losses	2	1	1	4	2	5	3	1	1
	Difference	0	+3	+4	-3	+1	-4	0	+3	+4
	Rank	4	2	2	5	3	6	4	2	2
MOEA/D	Wins	3	4	4	5	5	3	4	4	1
	Losses	2	1	2	1	1	2	1	1	3
	Difference	+1	+3	+2	+4	+4	+1	+3	+3	-2
	Rank	3	2	3	2	2	3	2	2	4
KnDE	Wins	5	3	0	0	2	0	1	0	0
	Losses	0	3	5	6	4	6	4	5	6
	Difference	+5	0	-5	-6	-2	-6	-3	-5	-6
	Rank	1	4	6	7	5	7	5	6	7
KnPSO	Wins	0	0	2	3	1	5	1	2	1
	Losses	4	6	4	3	5	1	4	4	3
	Difference	-4	-6	-2	0	-4	+4	-3	-2	-2
	Rank	6	7	5	4	6	2	5	5	4
SrEA/D	Wins	0	2	6	4	3	2	4	3	4
	Losses	3	4	0	2	2	2	1	3	2
	Difference	-3	-2	+6	+2	+1	0	+3	0	+2
	Rank	5	5	1	3	3	4	2	4	3
CDAS-SMPSO	Wins	0	1	3	1	0	2	0	0	1
	Losses	4	5	3	4	6	3	6	5	3
	Difference	-4	-4	0	-3	-6	-1	-6	-5	-2
	Rank	6	6	4	5	7	5	7	6	4
KnEA	Wins	5	6	0	6	6	6	6	6	6
	Losses	0	0	5	0	0	0	0	0	0
	Difference	+5	+6	-5	+6	+6	+6	+6	+6	+6
	Rank	1	1	6	1	1	1	1	1	1

Table 8.10: Hypervolume Rank Summary Using 30 Decision Variables, 10 Objectives

	Algorithm						
Measure	NSGA-III	MOEA/D	KnDE	KnPSO	SrEA/D	CDAS-SMPSO	KnEA
Mean	3.333	2.556	5.333	4.889	3.333	5.556	**1.556**
Standard Deviation	1.414	0.685	2.25	1.37	1.197	1.066	1.571
Maximum	6.0	4.0	7.0	7.0	6.0	7.0	6.0
Minimum	2.0	2.0	1.0	2.0	2.0	4.0	1.0

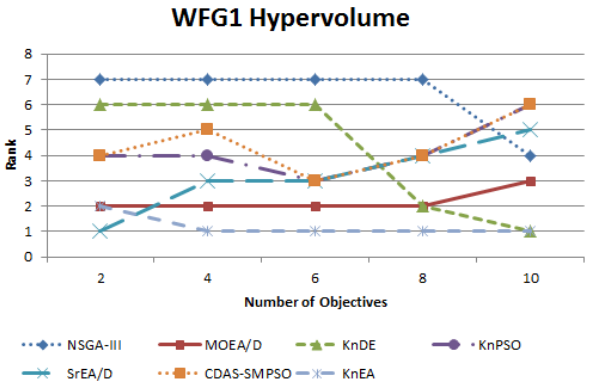

Figure 8.1: Ranking of hypervolume metric vs. number of objectives for the WFG1 problem

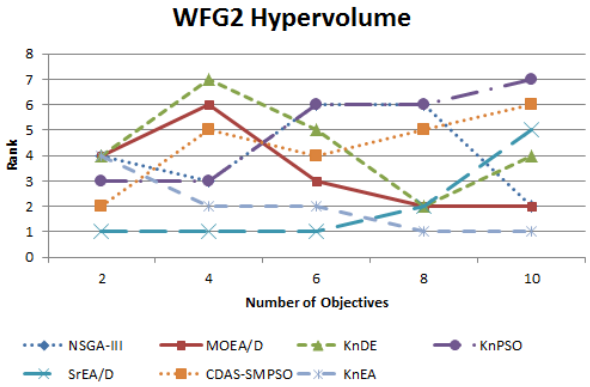

Figure 8.2: Ranking of hypervolume metric vs. number of objectives for the WFG2 problem

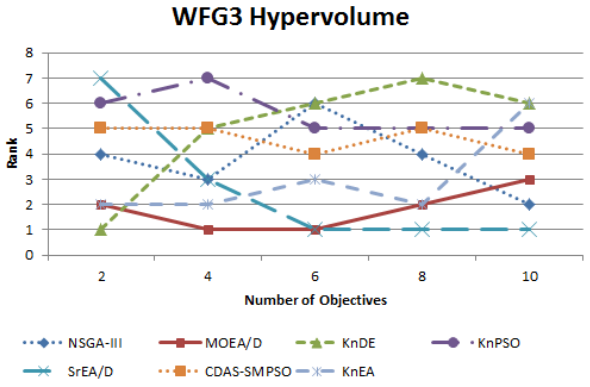

Figure 8.3: Ranking of hypervolume metric vs. number of objectives for the WFG3 problem

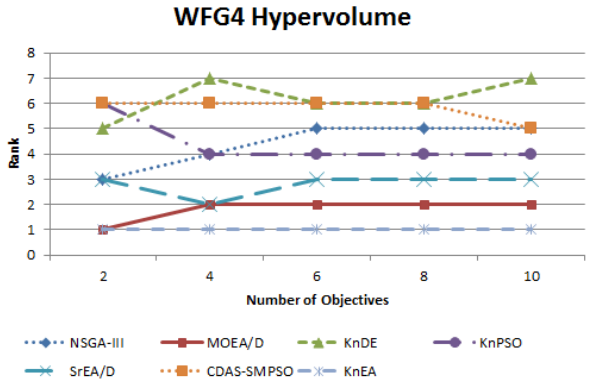

Figure 8.4: Ranking of hypervolume metric vs. number of objectives for the WFG4 problem

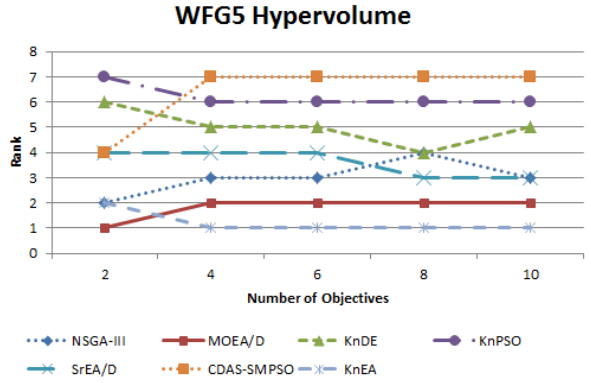

Figure 8.5: Ranking of hypervolume metric vs. number of objectives for the WFG5 problem

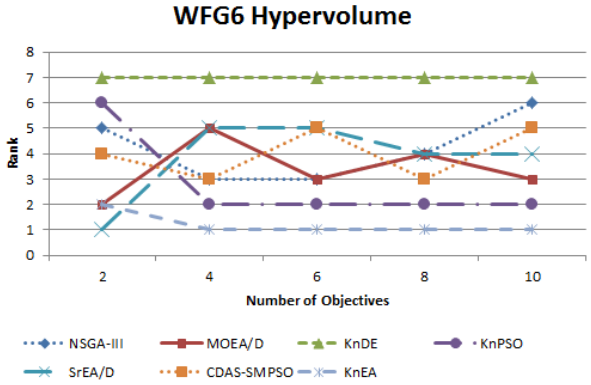

Figure 8.6: Ranking of hypervolume metric vs. number of objectives for the WFG6 problem

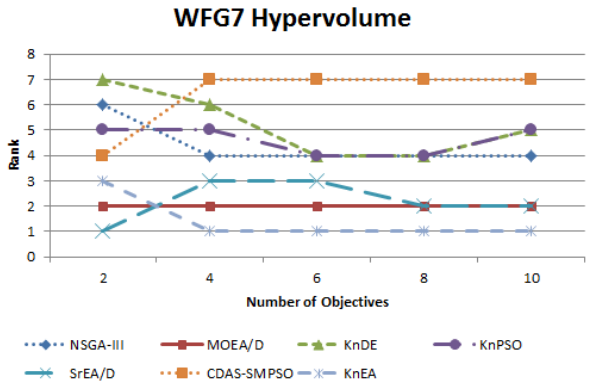

Figure 8.7: Ranking of hypervolume metric vs. number of objectives for the WFG7 problem

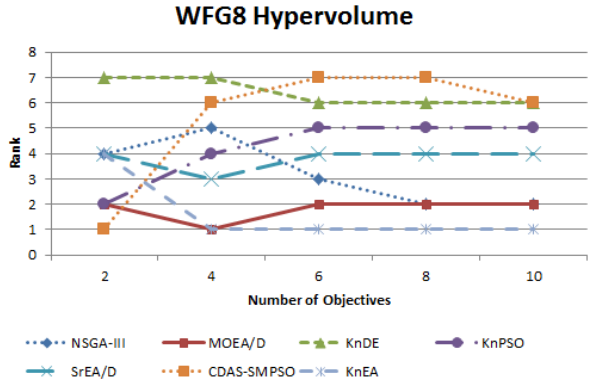

Figure 8.8: Ranking of hypervolume metric vs. number of objectives for the WFG8 problem

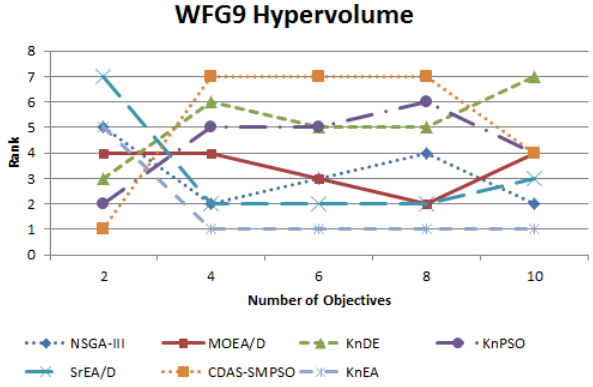

Figure 8.9: Ranking of hypervolume metric vs. number of objectives for the WFG9 problem

8.2 Inverted Generational Distance

IGD metric results over an increasing number of objectives are presented for each algorithm in Tables 8.11-8.21. For the bi-objective test instances in Table 8.12, the evolutionary knee-driven approach fared best, as KnDE and KnEA tied for the lowest mean ranking with a value of 2.889. We note that SrEA/D, MOEA/D and NSGA-III each possessed poor performance, indicating that the decomposition-based approach was ineffective for the low number of objectives encountered here. However, for test instances where the number of objectives was greater than two, the decomposition algorithms performed much more promisingly. Specifically, for the four, six, eight and ten objective cases in Tables 8.13-8.20, the SrEA/D algorithm possessed the best mean rank of 2.0, 2.222, 2.778 and 2.667, respectively. In these instances, SrEA/D produced a set of well-spread solutions that converged better to the Pareto-optimal front than the other tested CIMOOS. The MOEA/D algorithm scaled very well in general, as its mean rank increased from 4.778 for the four objective instances (Table 8.14) to 2.778 for the ten objective cases (Table 8.20). We also note that the maximum rank of MOEA/D was the best among all algorithms when six or more objectives were employed, which in combination with low standard deviation values is indicative of generally consistent performance. The NSGA-III algorithm was the worst performing decomposition-based optimizer when the number of objectives were high, possessing a mean rank of 4.556 for the ten objective MaOPs in Table 8.20.

Analyzing performance of the knee-driven algorithms, one will observe that the IGD metric performance of KnEA was much less satisfactory than its hypervolume performance in Section 8.1. KnEA obtained a desirable mean rank of 2.889 for the bi-objective instances in Table 8.12, however as the number of objectives increased the IGD metric performance of KnEA waned. When the number of objectives were increased to six, KnEA experienced significant performance degradation as evidenced by its poor mean rank of 4.778 in Table 8.16. From the difference in performance be-

tween the produced hypervolume and IGD metric values, we hypothesize that KnEA covers a larger region of the search, albeit with large gaps in the produced front. Performance of KnDE and KnPSO was also quite mediocre overall for all but the bi-objective MOPs. KnPSO experienced its worst mean rank of 5.111 for the four objective test instances in Table 8.14, whereas KnDE performed worst with an average ranking of 5.222 when eight objectives were employed in Table 8.18. However, for the ten objective MaOPs in Table 8.20, the mean rank of KnPSO improves to 3.778. KnPSO seems to exhibit the potential to scale well, thus we encourage further analysis on its scalability in instances where more than ten objective are employed.

In Figures 8.10-8.18, the rank of each algorithm is displayed for problems WFG1-WFG, respectively. Given in Figure 8.10, the KnEA algorithm performs best for the WFG1 function as the number of objectives increase. One should note that this observation was also present for the hypervolume experiments in Section 8.1, thus we conclude that KnEA is adept at converging well while simultaneously maintaining a desirable solution spread when presented with the flat, polynomial bias encountered in WFG1. The other evolutionary knee-driven algorithm, KnDE, seemed to experience significantly more difficulty for many problem instances, namely the WFG3, WFG4, WFG5 and WFG9 functions as evidenced by Figures 8.12, 8.13, 8.14 and 8.18, respectively. For the CDAS-SMPSO algorithm, Figures 8.10-8.18 carry a general trend in that mean ranking increases as the number of objectives grow larger. For problems WFG2, WFG4 and WFG6-9 CDAS-SMPSO possessed the worst rank overall when the number of objectives were ten, given in Figures 8.11, 8.13 and 8.15-8.18, respectively. Another notable observation is viewable in Figure 8.18, where the decomposition-based optimizers outperformed all other algorithms on the challenging WFG9 problem, which is non-separable and contains parameter dependencies.

Table 8.11: Mann-Whitney Wins and Losses For The IGD Metric Using 30 Decision Variables, 2 Objectives

		WFG Function								
Algorithm	**Result**	1	2	3	4	5	6	7	8	9
NSGA-III	Wins	1	3	3	5	0	1	3	3	1
	Losses	5	0	2	1	4	5	3	3	4
	Difference	-4	+3	+1	+4	-4	-4	0	0	-3
	Rank	6	3	3	2	5	6	4	4	6
MOEA/D	Wins	3	2	3	3	3	2	3	0	2
	Losses	1	3	2	2	3	3	2	6	2
	Difference	+2	-1	+1	+1	0	-1	+1	-6	0
	Rank	2	5	3	3	4	4	3	7	3
KnDE	Wins	6	2	6	2	5	5	5	4	0
	Losses	0	2	0	4	1	1	0	2	4
	Difference	+6	0	+6	-2	+4	+4	+5	+2	-4
	Rank	1	4	1	5	2	2	1	3	7
KnPSO	Wins	3	4	0	0	0	4	0	5	5
	Losses	1	0	6	6	4	2	6	1	1
	Difference	+2	+4	-6	-6	-4	+2	-6	+4	+4
	Rank	2	1	7	7	5	3	7	2	2
SrEA/D	Wins	0	0	2	3	0	0	2	6	0
	Losses	6	5	4	2	4	6	3	0	2
	Difference	-6	-5	-2	+1	-4	-6	-1	+6	-2
	Rank	7	6	5	3	5	7	5	1	5
CDAS-SMPSO	Wins	3	0	1	1	6	6	1	1	6
	Losses	1	5	5	5	0	0	5	4	0
	Difference	+2	-5	-4	-4	+6	+6	-4	-3	+6
	Rank	2	6	6	6	1	1	6	5	1
KnEA	Wins	2	4	5	6	4	2	5	1	2
	Losses	4	0	1	0	2	3	0	4	3
	Difference	-2	+4	+4	+6	+2	-1	+5	-3	-1
	Rank	5	1	2	1	3	4	1	5	4

Table 8.12: IGD Rank Summary Using 30 Decision Variables, 2 Objectives

	Algorithm						
Measure	NSGA-III	MOEA/D	KnDE	KnPSO	SrEA/D	CDAS-SMPSO	KnEA
Mean	4.333	3.778	**2.889**	4.0	4.889	3.778	**2.889**
Standard Deviation	1.414	1.397	1.969	2.357	1.792	2.299	1.595
Maximum	6.0	7.0	7.0	7.0	7.0	6.0	5.0
Minimum	2.0	2.0	1.0	1.0	1.0	1.0	1.0

Table 8.13: Mann-Whitney Wins and Losses For The IGD Metric Using 30 Decision Variables, 4 Objectives

		WFG Function								
Algorithm	**Result**	1	2	3	4	5	6	7	8	9
NSGA-III	Wins	0	1	5	0	4	3	5	0	2
	Losses	5	4	1	3	0	3	1	3	1
	Difference	-5	-3	+4	-3	+4	0	+4	-3	+1
	Rank	6	6	2	4	1	4	2	5	3
MOEA/D	Wins	6	1	4	0	1	0	1	0	0
	Losses	0	3	2	3	5	5	5	4	5
	Difference	+6	-2	+2	-3	-4	-5	-4	-4	-5
	Rank	1	5	3	4	6	6	6	6	6
KnDE	Wins	5	5	1	0	3	2	2	4	2
	Losses	1	1	5	3	0	4	2	1	2
	Difference	+4	+4	-4	-3	+3	-2	0	+3	0
	Rank	2	2	6	4	3	5	3	2	4
KnPSO	Wins	4	2	0	4	0	0	0	2	0
	Losses	2	2	6	1	6	5	6	3	5
	Difference	+2	0	-6	+3	-6	-5	-6	-1	-5
	Rank	3	4	7	2	7	6	7	4	6
SrEA/D	Wins	0	6	6	6	4	4	6	4	2
	Losses	5	0	0	0	0	0	0	1	2
	Difference	-5	+6	+6	+6	+4	+4	+6	+3	0
	Rank	6	1	1	1	1	1	1	2	4
CDAS-SMPSO	Wins	2	0	2	4	2	4	2	6	5
	Losses	4	6	4	1	2	0	2	0	0
	Difference	-2	-6	-2	+3	0	+4	0	+6	+5
	Rank	5	7	5	2	4	1	3	1	1
KnEA	Wins	3	3	3	0	2	4	2	0	4
	Losses	3	2	3	3	3	0	2	4	0
	Difference	0	+1	0	-3	-1	+4	0	-4	+4
	Rank	4	3	4	4	5	1	3	6	2

Table 8.14: IGD Rank Summary Using 30 Decision Variables, 4 Objectives

	Algorithm						
Measure	NSGA-III	MOEA/D	KnDE	KnPSO	SrEA/D	CDAS-SMPSO	KnEA
Mean	3.667	4.778	3.444	5.111	**2.0**	3.222	3.556
Standard Deviation	1.7	1.685	1.343	1.792	1.7	2.043	1.423
Maximum	6.0	6.0	6.0	7.0	6.0	7.0	6.0
Minimum	1.0	1.0	2.0	2.0	1.0	1.0	1.0

Table 8.15: Mann-Whitney Wins and Losses For The IGD Metric Using 30 Decision Variables, 6 Objectives

		WFG Function								
Algorithm	**Result**	1	2	3	4	5	6	7	8	9
NSGA-III	Wins	0	1	5	1	1	3	0	0	5
	Losses	6	5	0	1	4	2	2	6	0
	Difference	-6	-4	+5	0	-3	+1	-2	-6	+5
	Rank	7	6	1	4	6	4	4	7	1
MOEA/D	Wins	5	2	4	2	4	1	3	3	3
	Losses	1	3	2	3	1	4	2	2	1
	Difference	+4	-1	+2	-1	+3	-3	+1	+1	+2
	Rank	2	4	3	5	3	5	3	3	3
KnDE	Wins	4	4	0	0	2	0	1	2	1
	Losses	2	0	6	6	3	5	3	3	4
	Difference	+2	+4	-6	-6	-1	-5	-2	-1	-3
	Rank	3	1	7	7	4	7	4	5	5
KnPSO	Wins	3	2	1	4	2	4	1	2	0
	Losses	3	3	5	0	4	2	3	1	6
	Difference	0	-1	-4	+4	-2	+2	-2	+1	-6
	Rank	4	4	6	1	5	3	4	3	7
SrEA/D	Wins	1	4	5	4	4	4	6	4	3
	Losses	5	0	0	0	0	0	0	1	2
	Difference	-4	+4	+5	+4	+4	+4	+6	+3	+1
	Rank	6	1	1	1	2	2	1	2	4
CDAS-SMPSO	Wins	2	0	2	3	5	5	5	6	4
	Losses	4	6	4	2	0	0	1	0	0
	Difference	-2	-6	-2	+1	+5	+5	+4	+6	+4
	Rank	5	7	5	3	1	1	2	1	2
KnEA	Wins	6	4	3	2	0	0	0	1	1
	Losses	0	0	3	4	6	4	5	5	4
	Difference	+6	+4	0	-2	-6	-4	-5	-4	-3
	Rank	1	1	4	6	7	6	7	6	5

Table 8.16: IGD Rank Summary Using 30 Decision Variables, 6 Objectives

	Algorithm						
Measure	NSGA-III	MOEA/D	KnDE	KnPSO	SrEA/D	CDAS-SMPSO	KnEA
Mean	4.444	3.444	4.778	4.111	**2.222**	3.0	4.778
Standard Deviation	2.166	0.956	1.931	1.663	1.618	2.055	2.2
Maximum	7.0	5.0	7.0	7.0	6.0	7.0	7.0
Minimum	1.0	2.0	1.0	1.0	1.0	1.0	1.0

Table 8.17: Mann-Whitney Wins and Losses For The IGD Metric Using 30 Decision Variables, 8 Objectives

		WFG Function								
Algorithm	**Result**	1	2	3	4	5	6	7	8	9
NSGA-III	Wins	0	1	5	5	6	2	0	0	5
	Losses	5	4	1	0	0	3	3	6	0
	Difference	-5	-3	+4	+5	+6	-1	-3	-6	+5
	Rank	6	5	2	1	1	5	5	7	1
MOEA/D	Wins	5	1	4	1	5	5	6	2	4
	Losses	1	4	2	3	1	0	0	3	1
	Difference	+4	-3	+2	-2	+4	+5	+6	-1	+3
	Rank	2	5	3	5	2	1	1	4	3
KnDE	Wins	4	4	0	0	0	1	0	4	0
	Losses	2	1	6	6	5	5	5	0	5
	Difference	+2	+3	-6	-6	-5	-4	-5	+4	-5
	Rank	3	2	7	7	6	6	7	2	7
KnPSO	Wins	3	3	1	2	2	3	1	4	1
	Losses	3	3	5	3	4	3	4	1	3
	Difference	0	0	-4	-1	-2	0	-3	+3	-2
	Rank	4	4	6	4	5	4	5	3	5
SrEA/D	Wins	0	6	6	5	3	4	5	1	2
	Losses	5	0	0	0	2	0	1	3	3
	Difference	-5	+6	+6	+5	+1	+4	+4	-2	-1
	Rank	6	1	1	1	3	2	2	5	4
CDAS-SMPSO	Wins	2	0	2	3	3	0	4	5	4
	Losses	4	6	4	2	2	6	2	0	0
	Difference	-2	-6	-2	+1	+1	-6	+2	+5	+4
	Rank	5	7	5	3	3	7	3	1	2
KnEA	Wins	6	4	3	2	0	4	2	1	0
	Losses	0	1	3	4	5	2	3	4	4
	Difference	+6	+3	0	-2	-5	+2	-1	-3	-4
	Rank	1	2	4	5	6	3	4	6	6

Table 8.18: IGD Rank Summary Using 30 Decision Variables, 8 Objectives

	Algorithm						
Measure	NSGA-III	MOEA/D	KnDE	KnPSO	SrEA/D	CDAS-SMPSO	KnEA
Mean	3.667	2.889	5.222	4.444	**2.778**	4.0	4.111
Standard Deviation	2.261	1.449	2.096	0.831	1.75	2.0	1.728
Maximum	7.0	5.0	7.0	6.0	6.0	7.0	6.0
Minimum	1.0	1.0	2.0	3.0	1.0	1.0	1.0

Table 8.19: Mann-Whitney Wins and Losses For The IGD Metric Using 30 Decision Variables, 10 Objectives

		WFG Function								
Algorithm	**Result**	1	2	3	4	5	6	7	8	9
NSGA-III	Wins	0	1	4	1	2	1	0	0	6
	Losses	6	4	1	2	1	5	4	5	0
	Difference	-6	-3	+3	-1	+1	-4	-4	-5	+6
	Rank	7	5	2	5	3	6	6	6	1
MOEA/D	Wins	5	1	4	3	6	4	4	3	5
	Losses	1	4	1	3	0	1	1	3	1
	Difference	+4	-3	+3	0	+6	+3	+3	0	+4
	Rank	2	5	2	4	1	2	3	4	2
KnDE	Wins	4	4	0	1	0	2	1	6	1
	Losses	2	1	6	4	6	4	3	0	5
	Difference	+2	+3	-6	-3	-6	-2	-2	+6	-4
	Rank	3	2	7	6	7	5	4	1	6
KnPSO	Wins	3	4	1	3	2	3	1	5	2
	Losses	3	2	5	2	1	3	3	1	3
	Difference	0	+2	-4	+1	+1	0	-2	+4	-1
	Rank	4	3	6	3	3	4	4	2	5
SrEA/D	Wins	1	6	6	6	2	6	5	2	3
	Losses	5	0	0	0	2	0	0	4	2
	Difference	-4	+6	+6	+6	0	+6	+5	-2	+1
	Rank	6	1	1	1	5	1	1	5	3
CDAS-SMPSO	Wins	2	0	2	0	3	0	1	0	0
	Losses	4	6	4	6	1	6	5	5	6
	Difference	-2	-6	-2	-6	+2	-6	-4	-5	-6
	Rank	5	7	5	7	2	7	6	6	7
KnEA	Wins	6	3	3	4	1	4	5	4	2
	Losses	0	2	3	1	5	1	1	2	2
	Difference	+6	+1	0	+3	-4	+3	+4	+2	0
	Rank	1	4	4	2	6	2	2	3	4

Table 8.20: IGD Rank Summary Using 30 Decision Variables, 10 Objectives

	Algorithm						
Measure	NSGA-III	MOEA/D	KnDE	KnPSO	SrEA/D	CDAS-SMPSO	KnEA
Mean	4.556	2.778	4.556	3.778	**2.667**	5.778	3.111
Standard Deviation	1.95	1.227	2.061	1.133	2.0	1.548	1.449
Maximum	7.0	5.0	7.0	6.0	6.0	7.0	6.0
Minimum	1.0	1.0	1.0	2.0	1.0	2.0	1.0

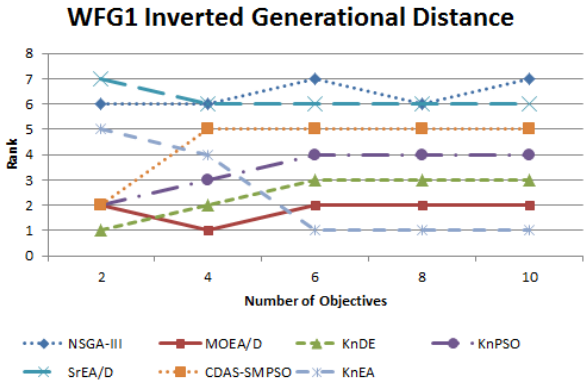

Figure 8.10: Ranking of IGD metric vs. number of objectives for the WFG1 problem

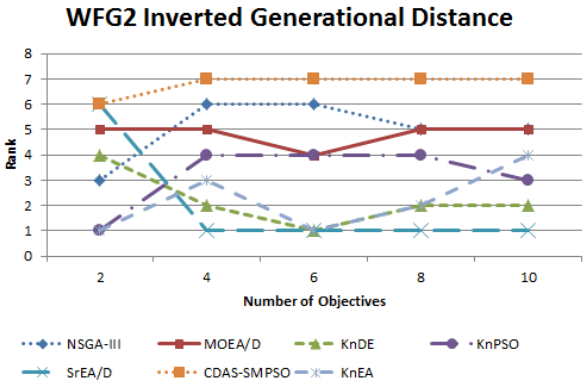

Figure 8.11: Ranking of IGD metric vs. number of objectives for the WFG2 problem

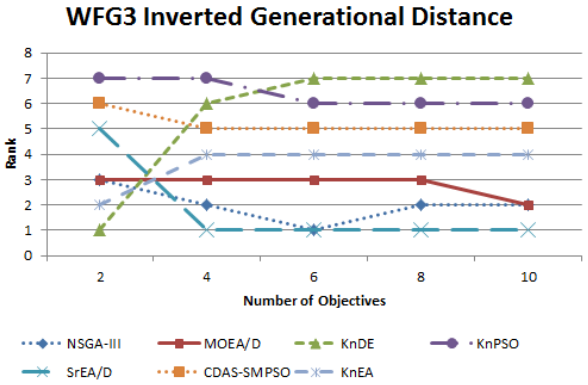

Figure 8.12: Ranking of IGD metric vs. number of objectives for the WFG3 problem

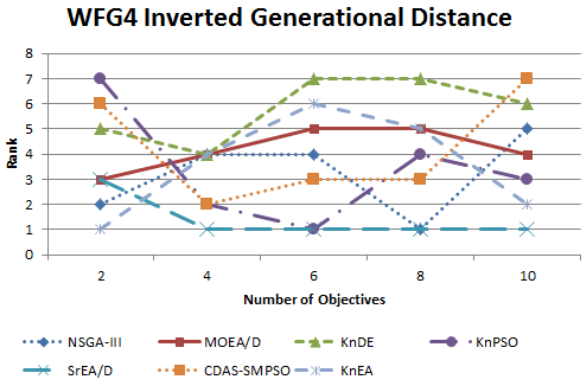

Figure 8.13: Ranking of IGD metric vs. number of objectives for the WFG4 problem

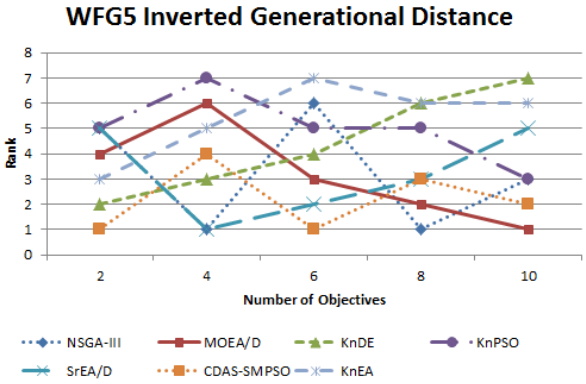

Figure 8.14: Ranking of IGD metric vs. number of objectives for the WFG5 problem

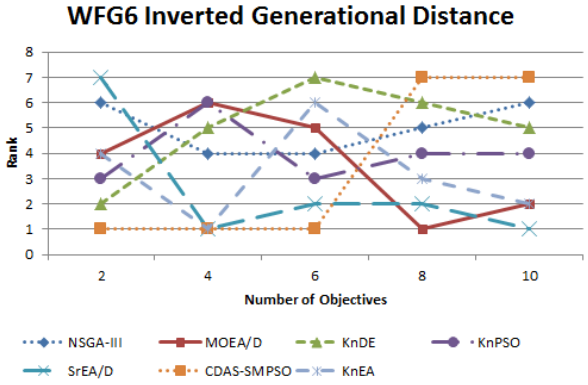

Figure 8.15: Ranking of IGD metric vs. number of objectives for the WFG6 problem

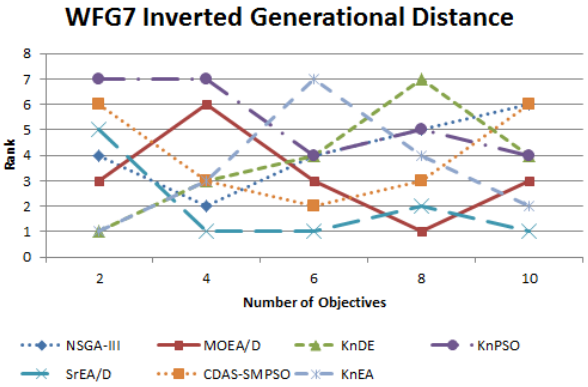

Figure 8.16: Ranking of IGD metric vs. number of objectives for the WFG7 problem

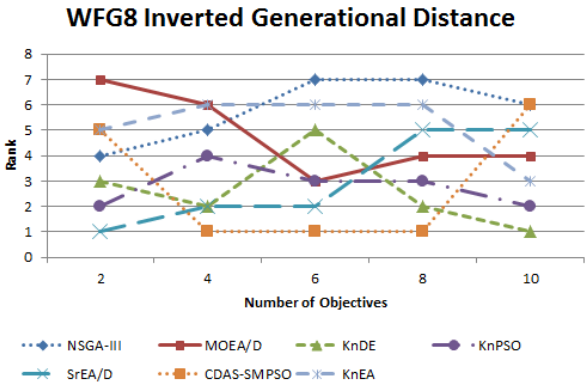

Figure 8.17: Ranking of IGD metric vs. number of objectives for the WFG8 problem

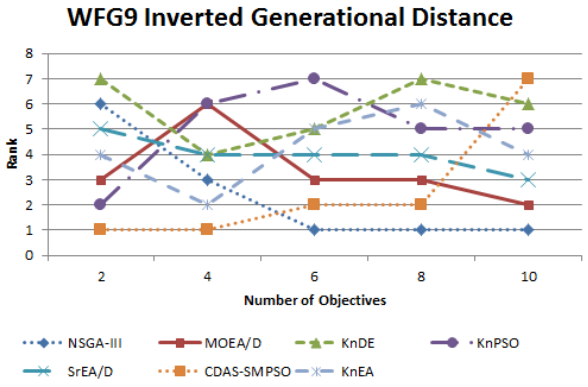

Figure 8.18: Ranking of IGD metric vs. number of objectives for the WFG9 problem

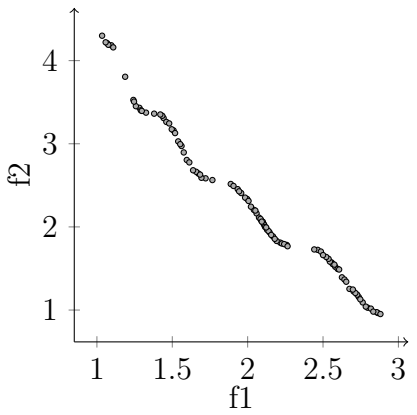

Figure 8.19: Sample Pareto front produced by CDAS-SMPSO on WFG1.

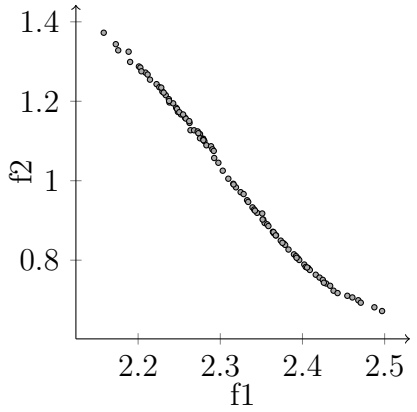

Figure 8.20: Sample Pareto front produced by NSGA-III on WFG1.

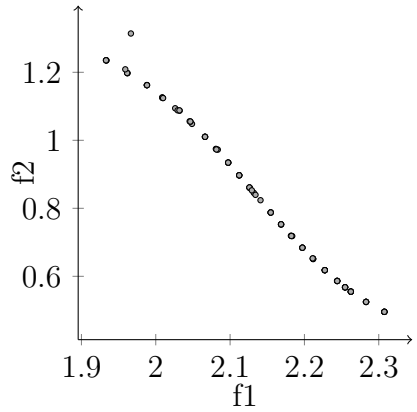

Figure 8.21: Sample Pareto front produced by MOEA/D on WFG1.

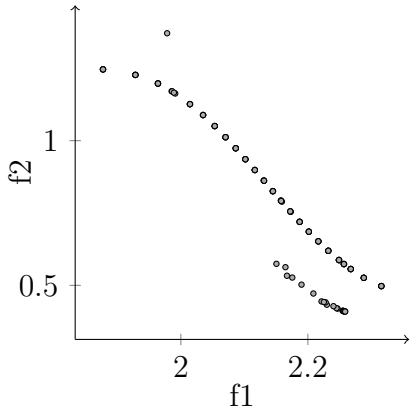

Figure 8.22: Sample Pareto front produced by SrEA/D on WFG1.

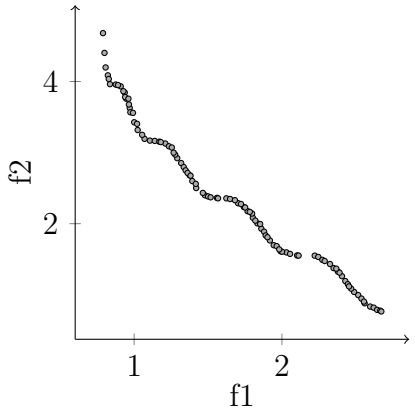

Figure 8.23: Sample Pareto front produced by KnDE on WFG1.

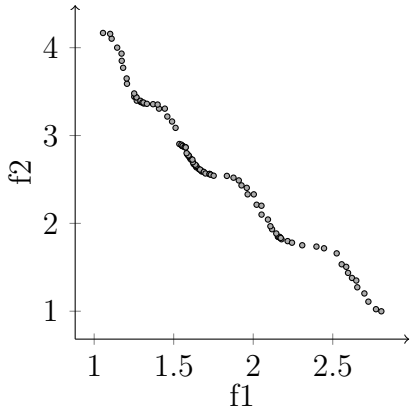

Figure 8.24: Sample Pareto front produced by KnPSO on WFG1.

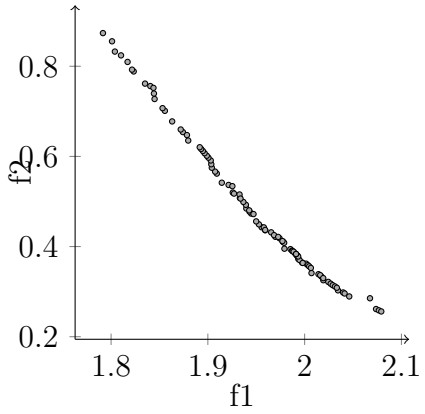

Figure 8.25: Sample Pareto front produced by KnEA on WFG1.

Chapter 9

Decision Variable Scalability Experiments

Within this chapter, the performance of each algorithm is analyzed as the number of decision variables increase. All experiments utilize 10 objectives, with the number of decision variables ranging from 30-1000. Section 9.1 consists of analyzing algorithm performance with regards to the hypervolume metric. Section 9.2 presents the results of the IGD metric experiments.

9.1 Hypervolume

Hypervolume performance of each algorithm over an increasing number of decision variables is displayed in Tables 9.1-9.8. When the number of decision variables are increased to 100 in Table 9.4, the mean rank of KnEA increases to 4.444, which is a significant increase in comparison to its average rank of 1.556 for the 30 decision variable instances in Table 9.2. This performance disparity is striking, indicating that KnEA experiences a significant loss of solution spread and/or convergence to the Pareto-optimal front for MaOPs when the number of decision variables are increased. Interestingly, MOEA/D and SrEA/D did not experience performance loss here, as MOEA/D performed best overall with a mean rank of 2.222 in Table 9.4. However,

when the number of decision variables were increased past 100, seen in Figures 9.5-9.8, we see that many of the evolutionary algorithms scaled quite poorly. The NSGA-III, KnDE and KnEA algorithms were the three worst performers overall, yielding mean ranks of 4.444, 5.889 and 6.111, respectively, for the 1000 decision variable instance in Table 9.8. Conversely, the PSO algorithms performed exceptionally well when the number of decision variables were 1000, with KnPSO and CDAS-SMPSO obtaining mean ranks of 2.333 and 1.778, respectively. These observations suggest that evolutionary approaches may perform worse than PSO approaches for MaOPs with large numbers of decision variables. The culprit is likely the crossover operation as mentioned in [27], since the sheer size of the search space causes two distant parents to produce a child that is distant from both of them. This problem is further compounded by the additional epistasis introduced as the number of decision variables grow. However, it is notable that the SrEA/D and MOEA/D algorithms did not scale as poorly as the other evolutionary optimizers, as SrEA/D possessed a formidable mean rank of 2.889 for the 500 decision variable instances in Table 9.6 and MOEA/D possessed a mean rank of 3.333 for the 1000 decision variable experiments in Table 9.8. Since both MOEA/D and SrEA/D employ a mating restriction scheme wherein solutions can only mate with other solutions in their neighbourhood, we conclude that this approach improved the effectiveness of the crossover operators for both algorithms. Mating restriction may be key for EAs to cope with the massive search space experienced for large-scale MaOPs, since the crossover operator will rarely mate distant parents together.

Figures 9.1-9.9 displays algorithmic rank vs number of decision variables for each of the nine WFG functions. For the WFG1 function in Figure 9.1, all EAs except MOEA/D scaled poorly as the number of objectives grew. On the non-separable WFG2 function, depicted in Figure 9.2, the KnPSO and NSGA-III algorithms yielded the best IGD performances as the number of objectives increased, while MOEA/D

experienced the worst scalability overall. The PSO algorithms, CDAS-SMPSO and KnPSO, possessed especially good performance for WFG1, WFG3, WFG6, WFG9, given in Figures 9.1, 9.3, 9.6 and 9.9, respectively. Regarding the knee-driven evolutionary optimizers, both KnDE and KnEA yielded the worst IGD ranks for the 1000 decision variable instances on the multi-modal WFG4 problem, deceptive WFG5 function, non-separable and reduced WFG6 problem, separable and uni-modal WFG7 problem, and non-separable parameter-dependent WFG8 function. SrEA/D performed notably well for WFG3, WFG4, WFG7 and WFG8.

Based on the conclusions drawn in this section, we make several recommendations for handling large-scale many-objective optimization:

1. When the number of decision variables increased, the evolutionary optimizers tended to performed much less satisfactory, with KnDE and KnEA obtaining the worst hypervolume performances overall. We attributed this to the reduced effectiveness of the crossover operator in large search spaces. However, since the MOEA/D and SrEA/D algorithms were able to scale better than the other EAs, we recommend the usage of a mating restriction scheme in order to maintain recombination operator effectiveness for large-scale MaOPs.

2. The swarm intelligence optimizers scaled very well in the face of increased decision variables, thus we encourage the usage of PSO algorithms for solving large-scale MaOPs. It would be interesting to extract the useful properties of the SrEA/D and MOEA/D EAs, which performed well in this section, into a PSO approach. One should note that the NF_{sum} method of SrEA/D has already been combined into a PSO algorithm by Ross *et. al* [4], referred to as the sum-of-ranks PSO (SrPSO), which would be an excellent candidate for solving large-scale MaOPs based on the conclusions in this section. Adding decomposition into SrPSO in a manner similar to SrEA/D would be interesting, perhaps improving performance even further.

Table 9.1: Mann-Whitney Wins and Losses For The Hypervolume Metric Using 30 Decision Variables, 10 Objectives

		WFG Function								
Algorithm	**Result**	1	2	3	4	5	6	7	8	9
NSGA-III	Wins	2	4	5	1	3	1	3	4	5
	Losses	2	1	1	4	2	5	3	1	1
	Difference	0	+3	+4	-3	+1	-4	0	+3	+4
	Rank	4	2	2	5	3	6	4	2	2
MOEA/D	Wins	3	4	4	5	5	3	4	4	1
	Losses	2	1	2	1	1	2	1	1	3
	Difference	+1	+3	+2	+4	+4	+1	+3	+3	-2
	Rank	3	2	3	2	2	3	2	2	4
KnDE	Wins	5	3	0	0	2	0	1	0	0
	Losses	0	3	5	6	4	6	4	5	6
	Difference	+5	0	-5	-6	-2	-6	-3	-5	-6
	Rank	1	4	6	7	5	7	5	6	7
KnPSO	Wins	0	0	2	3	1	5	1	2	1
	Losses	4	6	4	3	5	1	4	4	3
	Difference	-4	-6	-2	0	-4	+4	-3	-2	-2
	Rank	6	7	5	4	6	2	5	5	4
SrEA/D	Wins	0	2	6	4	3	2	4	3	4
	Losses	3	4	0	2	2	2	1	3	2
	Difference	-3	-2	+6	+2	+1	0	+3	0	+2
	Rank	5	5	1	3	3	4	2	4	3
CDAS-SMPSO	Wins	0	1	3	1	0	2	0	0	1
	Losses	4	5	3	4	6	3	6	5	3
	Difference	-4	-4	0	-3	-6	-1	-6	-5	-2
	Rank	6	6	4	5	7	5	7	6	4
KnEA	Wins	5	6	0	6	6	6	6	6	6
	Losses	0	0	5	0	0	0	0	0	0
	Difference	+5	+6	-5	+6	+6	+6	+6	+6	+6
	Rank	1	1	6	1	1	1	1	1	1

Table 9.2: Hypervolume Rank Summary Using 30 Decision Variables, 10 Objectives

	Algorithm						
Measure	NSGA-III	MOEA/D	KnDE	KnPSO	SrEA/D	CDAS-SMPSO	KnEA
Mean	3.333	2.556	5.333	4.889	3.333	5.556	**1.556**
Standard Deviation	1.414	0.685	2.25	1.37	1.197	1.066	1.571
Maximum	6.0	4.0	7.0	7.0	6.0	7.0	6.0
Minimum	2.0	2.0	1.0	2.0	2.0	4.0	1.0

Table 9.3: Mann-Whitney Wins and Losses For The Hypervolume Metric Using 100 Decision Variables, 10 Objectives

		WFG Function								
Algorithm	**Result**	1	2	3	4	5	6	7	8	9
NSGA-III	Wins	1	3	5	0	3	1	0	1	5
	Losses	5	0	0	5	2	4	4	2	0
	Difference	-4	+3	+5	-5	+1	-3	-4	-1	+5
	Rank	6	2	1	6	3	5	5	4	1
MOEA/D	Wins	6	1	1	6	6	4	6	6	3
	Losses	0	4	4	0	0	2	0	0	1
	Difference	+6	-3	-3	+6	+6	+2	+6	+6	+2
	Rank	1	5	5	1	1	3	1	1	2
KnDE	Wins	3	0	0	3	0	0	3	1	2
	Losses	1	6	6	1	5	6	2	3	3
	Difference	+2	-6	-6	+2	-5	-6	+1	-2	-1
	Rank	2	7	7	2	6	7	3	5	5
KnPSO	Wins	3	1	3	3	2	6	3	3	1
	Losses	1	4	3	1	4	0	2	2	5
	Difference	+2	-3	0	+2	-2	+6	+1	+1	-4
	Rank	2	5	4	2	5	1	3	3	6
SrEA/D	Wins	0	5	5	2	5	1	5	5	3
	Losses	6	0	0	4	1	4	1	1	1
	Difference	-6	+5	+5	-2	+4	-3	+4	+4	+2
	Rank	7	1	1	5	2	5	2	2	2
CDAS-SMPSO	Wins	3	3	4	3	0	5	0	1	0
	Losses	1	1	2	1	5	1	4	3	6
	Difference	+2	+2	+2	+2	-5	+4	-4	-2	-6
	Rank	2	3	3	2	6	2	5	5	7
KnEA	Wins	2	3	1	0	3	3	0	0	2
	Losses	4	1	4	5	2	3	4	6	0
	Difference	-2	+2	-3	-5	+1	0	-4	-6	+2
	Rank	5	3	5	6	3	4	5	7	2

Table 9.4: Hypervolume Rank Summary Using 100 Decision Variables, 10 Objectives

	Algorithm						
Measure	NSGA-III	MOEA/D	KnDE	KnPSO	SrEA/D	CDAS-SMPSO	KnEA
Mean	3.667	**2.222**	4.889	3.444	3.0	3.889	4.444
Standard Deviation	1.886	1.618	1.969	1.571	2.0	1.792	1.499
Maximum	6.0	5.0	7.0	6.0	7.0	7.0	7.0
Minimum	1.0	1.0	2.0	1.0	1.0	2.0	2.0

Table 9.5: Mann-Whitney Wins and Losses For The Hypervolume Metric Using 500 Decision Variables, 10 Objectives

		WFG Function								
Algorithm	**Result**	1	2	3	4	5	6	7	8	9
NSGA-III	Wins	1	6	3	2	2	2	3	3	1
	Losses	5	0	2	4	4	4	3	3	5
	Difference	-4	+6	+1	-2	-2	-2	0	0	-4
	Rank	6	1	3	5	5	5	4	4	6
MOEA/D	Wins	4	1	1	3	6	4	4	4	4
	Losses	0	5	5	2	0	2	2	1	2
	Difference	+4	-4	-4	+1	+6	+2	+2	+3	+2
	Rank	1	6	6	3	1	3	3	2	3
KnDE	Wins	3	0	0	1	1	0	1	1	3
	Losses	3	6	6	5	5	5	5	5	3
	Difference	0	-6	-6	-4	-4	-5	-4	-4	0
	Rank	4	7	7	6	6	6	6	6	4
KnPSO	Wins	4	5	3	3	3	6	2	2	6
	Losses	0	1	2	2	3	0	4	4	0
	Difference	+4	+4	+1	+1	0	+6	-2	-2	+6
	Rank	1	2	3	3	4	1	5	5	1
SrEA/D	Wins	0	4	5	6	4	3	6	6	2
	Losses	6	2	1	0	1	3	0	0	4
	Difference	-6	+2	+4	+6	+3	0	+6	+6	-2
	Rank	7	3	2	1	2	4	1	1	5
CDAS-SMPSO	Wins	4	3	6	5	4	5	5	4	5
	Losses	0	3	0	1	1	1	1	1	1
	Difference	+4	0	+6	+4	+3	+4	+4	+3	+4
	Rank	1	4	1	2	2	2	2	2	2
KnEA	Wins	2	2	2	0	0	0	0	0	0
	Losses	4	4	4	6	6	5	6	6	6
	Difference	-2	-2	-2	-6	-6	-5	-6	-6	-6
	Rank	5	5	5	7	7	6	7	7	7

Table 9.6: Hypervolume Rank Summary Using 500 Decision Variables, 10 Objectives

	Algorithm						
Measure	NSGA-III	MOEA/D	KnDE	KnPSO	SrEA/D	CDAS-SMPSO	KnEA
Mean	4.333	3.111	5.778	2.778	2.889	**2.0**	6.222
Standard Deviation	1.491	1.728	1.03	1.548	1.969	0.816	0.916
Maximum	6.0	6.0	7.0	5.0	7.0	4.0	7.0
Minimum	1.0	1.0	4.0	1.0	1.0	1.0	5.0

Table 9.7: Mann-Whitney Wins and Losses For The Hypervolume Metric Using 1000 Decision Variables, 10 Objectives

		WFG Function								
Algorithm	**Result**	1	2	3	4	5	6	7	8	9
NSGA-III	Wins	1	4	3	2	2	2	2	5	1
	Losses	5	1	3	4	4	4	4	1	5
	Difference	-4	+3	0	-2	-2	-2	-2	+4	-4
	Rank	6	2	4	5	5	5	5	2	6
MOEA/D	Wins	4	1	1	3	6	4	4	3	4
	Losses	0	5	5	2	0	2	2	3	2
	Difference	+4	-4	-4	+1	+6	+2	+2	0	+2
	Rank	1	6	6	3	1	3	3	4	3
KnDE	Wins	3	0	0	1	1	1	0	0	3
	Losses	3	6	6	5	5	5	6	5	3
	Difference	0	-6	-6	-4	-4	-4	-6	-5	0
	Rank	4	7	7	6	6	6	7	6	4
KnPSO	Wins	4	6	5	3	4	6	3	2	5
	Losses	0	0	1	2	2	0	3	4	0
	Difference	+4	+6	+4	+1	+2	+6	0	-2	+5
	Rank	1	1	2	3	3	1	4	5	1
SrEA/D	Wins	0	3	4	5	3	3	6	6	2
	Losses	6	2	2	1	3	3	0	0	4
	Difference	-6	+1	+2	+4	0	0	+6	+6	-2
	Rank	7	4	3	2	4	4	1	1	5
CDAS-SMPSO	Wins	4	3	6	6	5	5	5	4	5
	Losses	0	1	0	0	1	1	1	2	0
	Difference	+4	+2	+6	+6	+4	+4	+4	+2	+5
	Rank	1	3	1	1	2	2	2	3	1
KnEA	Wins	2	2	2	0	0	0	1	0	0
	Losses	4	4	4	6	6	6	5	5	6
	Difference	-2	-2	-2	-6	-6	-6	-4	-5	-6
	Rank	5	5	5	7	7	7	6	6	7

Table 9.8: Hypervolume Rank Summary Using 1000 Decision Variables, 10 Objectives

	Algorithm						
Measure	NSGA-III	MOEA/D	KnDE	KnPSO	SrEA/D	CDAS-SMPSO	KnEA
Mean	4.444	3.333	5.889	2.333	3.444	**1.778**	6.111
Standard Deviation	1.423	1.7	1.1	1.414	1.832	0.786	0.875
Maximum	6.0	6.0	7.0	5.0	7.0	3.0	7.0
Minimum	2.0	1.0	4.0	1.0	1.0	1.0	5.0

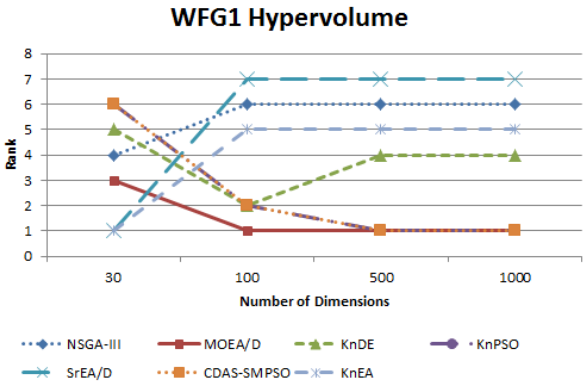

Figure 9.1: Ranking of hypervolume metric vs. number of decision variables for the WFG1 problem

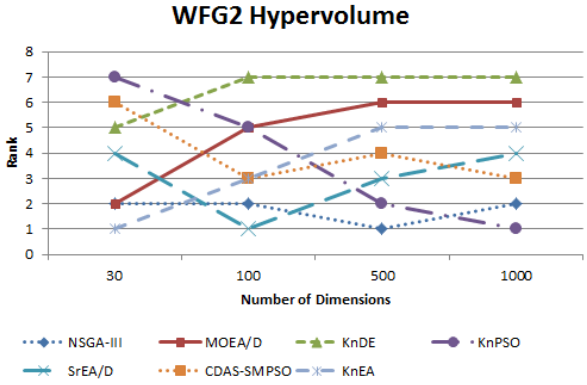

Figure 9.2: Ranking of hypervolume metric vs. number of decision variables for the WFG2 problem

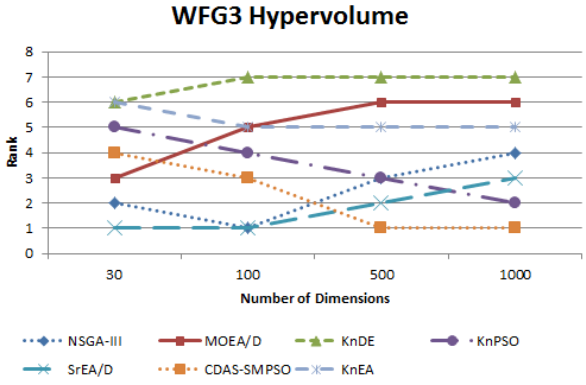

Figure 9.3: Ranking of hypervolume metric vs. number of decision variables for the WFG3 problem

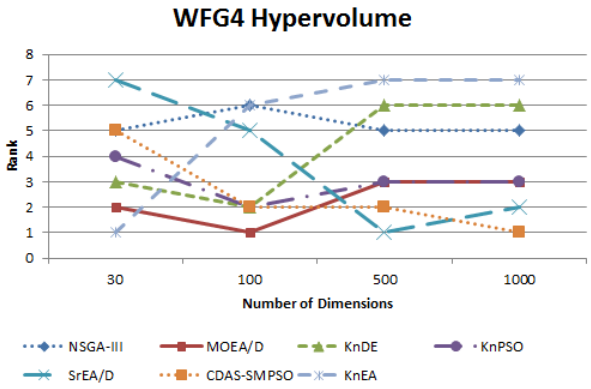

Figure 9.4: Ranking of hypervolume metric vs. number of decision variables for the WFG4 problem

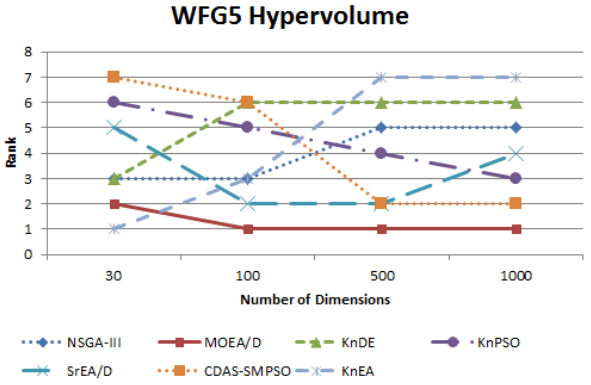

Figure 9.5: Ranking of hypervolume metric vs. number of decision variables for the WFG5 problem

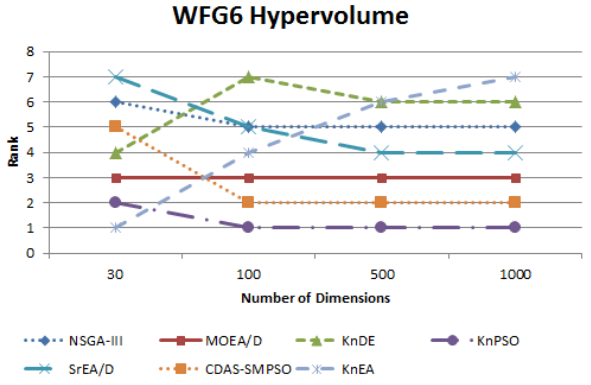

Figure 9.6: Ranking of hypervolume metric vs. number of decision variables for the WFG6 problem

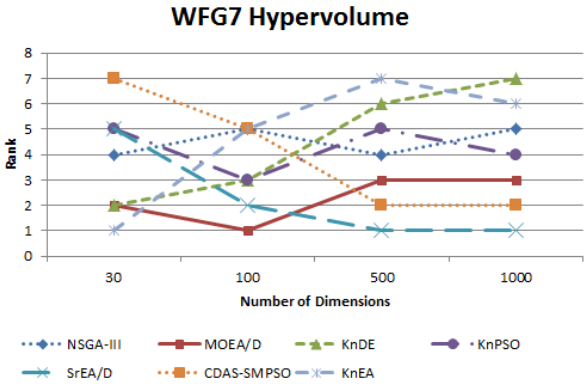

Figure 9.7: Ranking of hypervolume metric vs. number of decision variables for the WFG7 problem

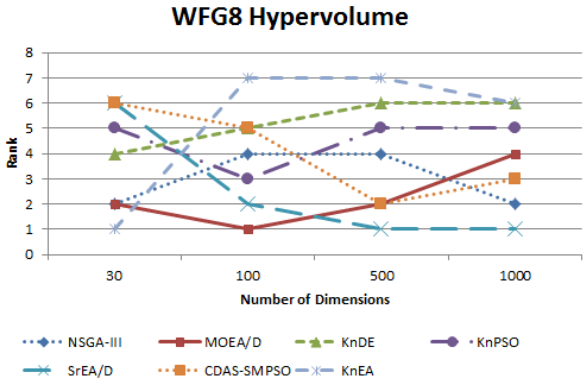

Figure 9.8: Ranking of hypervolume metric vs. number of decision variables for the WFG8 problem

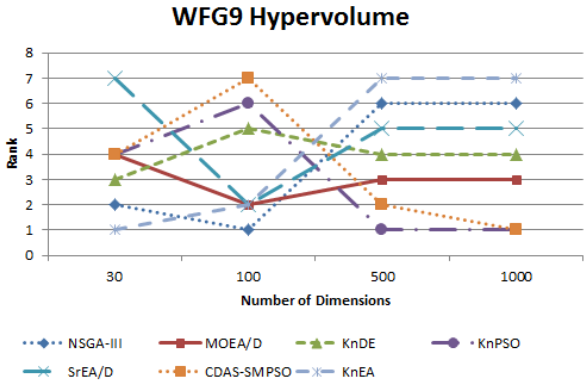

Figure 9.9: Ranking of hypervolume metric vs. number of decision variables for the WFG9 problem

9.2 Inverted Generational Distance

Tables 9.9-9.16 present an overview of the IGD metric performance for each algorithm as the number of decision variables increase. For the 30 and 100 decision variable test instances in Tables 9.10 and 9.12, respectively, SrEA/D obtains a better mean rank than all other algorithms. It is notable that the mean rank of SrEA/D decreases from 2.667 to 2.222 when the decision variables are increased to 100, whereas three of the other four EAs experience an increase in mean rank. The other EA whose mean rank does not increase, NSGA-III, also scales well to the 100 decision variable instances, decreasing its mean rank by 1.667. The performance of the knee-driven evolutionary approaches in Table 9.11 is poor, as KnDE fails to outperform any algorithms for WFG3, WFG4 and WFG7, and KnEA is outperformed by all other algorithms for WFG5 and WFG9. Regarding the PSO approaches, CDAS-SMPSO scales well with a decrease of 1.778 ranks on average, whereas KnPSO increases its mean rank slightly.

For the 500 and 1000 decision variable test instances, given in Figures 9.13 and 9.15, respectively, both KnPSO and CDAS-SMPSO tend to obtain a solution set that is better converged and more ideally distributed than the other tested CIMOOs. When 500 decision variables were employed, CDAS-SMPSO and KnPSO tied for the best IGD performance with a mean rank of 2.556, displayed in Table 9.14. When the number of decision variables were set to 1000, KnPSO outperformed all other algorithms with a mean rank of 2.0 on average. Overall, KnPSO scales quite well when the number of decision variables increase, especially considering that it possessed a mean rank of 4.889 for the 30 decision variable instances in Table 9.2. Combined with the high level of hypervolume performance observed in Section 9.1, it is clear that KnPSO performs very well when tasked with solving large-scale MaOPs. KnPSO seems to produce a set of well-converged, well-spread solutions which cover a considerable portion of the objective space. Here, we also note the low standard deviation of KnPSO on the 1000 decision variable instances, given as 1.155 in Table 9.16, which

indicates relatively consistent performance. Observing the performance of the evolutionary algorithms, NSGA-III was the best EA for the 500 decision variable test instance in Table 9.14 with a mean rank of 3.333, while SrEA/D yielded better IGD values than the other evolutionary optimizers when the number of decision variables were 1000 (Table 9.16). The other decomposition-based optimizer, MOEA/D, scaled poorly in terms of IGD as it increased its mean rank from 2.778 in Table 9.10 to 4.111 in Table 9.16. Regarding the knee-driven EAs, KnEA and KnDE were the two overall worst performing algorithms, possessing the lowest mean ranks in Tables 9.12, 9.14 and 9.16. Neither algorithm scaled as well as KnPSO when tasked with increasing numbers of decision variables. KnDE seemed to experience significant difficulties in producing a well-distributed, converging solution set as it was consistently outperformed by all other algorithms for 5 of the 9 WFG functions. KnDE was the overall worst performing optimizer, possessing the lowest mean rank of 5.667, 5.778 and 5.889 in Tables 9.12, 9.14 and 9.16, respectively.

The IGD rank of each tested optimizer using 30, 100, 500 and 1000 decision variables is displayed in Figures 9.10-9.18 for benchmark problems WFG1-WFG9. First and foremost, we note the consistency of KnPSO, which ranked second or better for the 1000 decision variable cases on all WFG functions except WFG7 (Figure 9.16) and WFG8 (Figure 9.17). For the WFG1 function, the MOEA/D and KnPSO algorithm scaled best overall to the increasing number of decision variables. This observation is consistent with Section 9.1, thus we conclude that both MOEA/D and KnPSO are superior to the other algorithms for the flat bias and mixed Pareto-optimal front geometries encountered in WFG1 when large amounts of decision variables (500+) are employed. Another notable observation is seen for the non-separable and reduced WFG6 function in Figure 9.15, where CDAS-SMPSO ranks worst in every instance and SrEA/D performs best in all cases. Since WFG6 is the only function where this phenomena occurs, it is likely that reduced, non-separable MaOPs present

CDAS-SMPSO with significant difficulties, but are easily solved by SrEA/D. Note that CDAS-SMPSO performs best overall in many other instances, namely for the WFG3, WFG4, WFG7 and WFG8 functions, given in Figures 9.12, 9.13, 9.16 and 9.17, respectively. Regarding the decomposition-based NSGA-III algorithm, notably good IGD ranks were observed for the deceptive WFG5 problem in Figure 9.14 and the separable but uni-modal WFG7 function in Figure 9.16. The evolutionary knee-driven optimizers, KnDE and KnEA, scaled quite poorly in many instances. KnDE possessed the worst mean rank for the 1000 decision variable cases on the WFG2, WFG3, WFG4, WFG7 and WFG8 functions, whereas KnEA struggled on WFG4, WFG5 and WFG9.

Table 9.9: Mann-Whitney Wins and Losses For The IGD Metric Using 30 Decision Variables, 10 Objectives

		WFG Function								
Algorithm	**Result**	1	2	3	4	5	6	7	8	9
NSGA-III	Wins	0	1	4	1	2	1	0	0	6
	Losses	6	4	1	2	1	5	4	5	0
	Difference	-6	-3	+3	-1	+1	-4	-4	-5	+6
	Rank	7	5	2	5	3	6	6	6	1
MOEA/D	Wins	5	1	4	3	6	4	4	3	5
	Losses	1	4	1	3	0	1	1	3	1
	Difference	+4	-3	+3	0	+6	+3	+3	0	+4
	Rank	2	5	2	4	1	2	3	4	2
KnDE	Wins	4	4	0	1	0	2	1	6	1
	Losses	2	1	6	4	6	4	3	0	5
	Difference	+2	+3	-6	-3	-6	-2	-2	+6	-4
	Rank	3	2	7	6	7	5	4	1	6
KnPSO	Wins	3	4	1	3	2	3	1	5	2
	Losses	3	2	5	2	1	3	3	1	3
	Difference	0	+2	-4	+1	+1	0	-2	+4	-1
	Rank	4	3	6	3	3	4	4	2	5
SrEA/D	Wins	1	6	6	6	2	6	5	2	3
	Losses	5	0	0	0	2	0	0	4	2
	Difference	-4	+6	+6	+6	0	+6	+5	-2	+1
	Rank	6	1	1	1	5	1	1	5	3
CDAS-SMPSO	Wins	2	0	2	0	3	0	1	0	0
	Losses	4	6	4	6	1	6	5	5	6
	Difference	-2	-6	-2	-6	+2	-6	-4	-5	-6
	Rank	5	7	5	7	2	7	6	6	7
KnEA	Wins	6	3	3	4	1	4	5	4	2
	Losses	0	2	3	1	5	1	1	2	2
	Difference	+6	+1	0	+3	-4	+3	+4	+2	0
	Rank	1	4	4	2	6	2	2	3	4

Table 9.10: IGD Rank Summary Using 30 Decision Variables, 10 Objectives

	Algorithm						
Measure	NSGA-III	MOEA/D	KnDE	KnPSO	SrEA/D	CDAS-SMPSO	KnEA
Mean	4.556	2.778	4.556	3.778	**2.667**	5.778	3.111
Standard Deviation	1.95	1.227	2.061	1.133	2.0	1.548	1.449
Maximum	7.0	5.0	7.0	6.0	6.0	7.0	6.0
Minimum	1.0	1.0	1.0	2.0	1.0	2.0	1.0

Table 9.11: Mann-Whitney Wins and Losses For The IGD Metric Using 100 Decision Variables, 10 Objectives

		WFG Function								
Algorithm	**Result**	1	2	3	4	5	6	7	8	9
NSGA-III	Wins	1	1	5	5	6	5	3	0	6
	Losses	5	2	0	1	0	0	1	6	0
	Difference	-4	-1	+5	+4	+6	+5	+2	-6	+6
	Rank	6	5	1	2	1	1	2	7	1
MOEA/D	Wins	6	2	1	2	4	1	3	5	4
	Losses	0	2	5	4	1	5	1	0	1
	Difference	+6	0	-4	-2	+3	-4	+2	+5	+3
	Rank	1	3	6	5	2	6	2	1	2
KnDE	Wins	5	1	0	0	1	2	0	1	1
	Losses	1	4	6	6	5	4	6	4	5
	Difference	+4	-3	-6	-6	-4	-2	-6	-3	-4
	Rank	2	6	7	7	6	5	7	5	6
KnPSO	Wins	4	2	2	4	2	3	1	1	2
	Losses	2	2	4	2	4	2	4	4	4
	Difference	+2	0	-2	+2	-2	+1	-3	-3	-2
	Rank	3	3	5	3	5	3	5	5	5
SrEA/D	Wins	0	6	5	6	4	5	6	4	3
	Losses	6	0	0	0	1	0	0	0	3
	Difference	-6	+6	+5	+6	+3	+5	+6	+4	0
	Rank	7	1	1	1	2	1	1	2	4
CDAS-SMPSO	Wins	3	0	3	3	3	0	3	4	4
	Losses	3	6	2	3	3	6	1	1	1
	Difference	0	-6	+1	0	0	-6	+2	+3	+3
	Rank	4	7	3	4	4	7	2	3	2
KnEA	Wins	2	5	3	1	0	3	1	3	0
	Losses	4	1	2	5	6	2	4	3	6
	Difference	-2	+4	+1	-4	-6	+1	-3	0	-6
	Rank	5	2	3	6	7	3	5	4	7

Table 9.12: IGD Rank Summary Using 100 Decision Variables, 10 Objectives

	Algorithm						
Measure	NSGA-III	MOEA/D	KnDE	KnPSO	SrEA/D	CDAS-SMPSO	KnEA
Mean	2.889	3.111	5.667	4.111	**2.222**	4.0	4.667
Standard Deviation	2.283	1.912	1.491	0.994	1.931	1.764	1.7
Maximum	7.0	6.0	7.0	5.0	7.0	7.0	7.0
Minimum	1.0	1.0	2.0	3.0	1.0	2.0	2.0

Table 9.13: Mann-Whitney Wins and Losses For The IGD Metric Using 500 Decision Variables, 10 Objectives

		WFG Function								
Algorithm	**Result**	1	2	3	4	5	6	7	8	9
NSGA-III	Wins	1	5	3	4	6	5	6	0	1
	Losses	5	1	2	2	0	1	0	5	5
	Difference	-4	+4	+1	+2	+6	+4	+6	-5	-4
	Rank	6	2	3	3	1	2	1	6	6
MOEA/D	Wins	6	1	1	2	2	3	3	4	4
	Losses	0	5	5	4	1	2	3	2	2
	Difference	+6	-4	-4	-2	+1	+1	0	+2	+2
	Rank	1	6	6	5	3	3	4	3	3
KnDE	Wins	3	0	0	0	1	2	0	0	3
	Losses	2	6	6	6	5	4	6	5	3
	Difference	+1	-6	-6	-6	-4	-2	-6	-5	0
	Rank	3	7	7	7	6	5	7	6	4
KnPSO	Wins	5	6	5	5	2	3	1	2	6
	Losses	1	0	1	1	1	2	4	3	0
	Difference	+4	+6	+4	+4	+1	+1	-3	-1	+6
	Rank	2	1	2	2	3	3	5	4	1
SrEA/D	Wins	0	4	3	3	2	6	4	5	2
	Losses	6	2	2	3	2	0	1	0	4
	Difference	-6	+2	+1	0	0	+6	+3	+5	-2
	Rank	7	3	3	4	5	1	2	1	5
CDAS-SMPSO	Wins	3	2	6	6	3	0	4	5	5
	Losses	2	3	0	0	1	6	1	0	1
	Difference	+1	-1	+6	+6	+2	-6	+3	+5	+4
	Rank	3	4	1	1	2	7	2	1	2
KnEA	Wins	2	2	2	1	0	1	1	2	0
	Losses	4	3	4	5	6	5	4	3	6
	Difference	-2	-1	-2	-4	-6	-4	-3	-1	-6
	Rank	5	4	5	6	7	6	5	4	7

Table 9.14: IGD Rank Summary Using 500 Decision Variables, 10 Objectives

	Algorithm						
Measure	NSGA-III	MOEA/D	KnDE	KnPSO	SrEA/D	CDAS-SMPSO	KnEA
Mean	3.333	3.778	5.778	**2.556**	3.444	**2.556**	5.444
Standard Deviation	2.0	1.548	1.397	1.257	1.892	1.832	1.066
Maximum	6.0	6.0	7.0	5.0	7.0	7.0	7.0
Minimum	1.0	1.0	3.0	1.0	1.0	1.0	4.0

Table 9.15: Mann-Whitney Wins and Losses For The IGD Metric Using 1000 Decision Variables, 10 Objectives

		WFG Function								
Algorithm	**Result**	1	2	3	4	5	6	7	8	9
NSGA-III	Wins	1	5	3	4	6	3	4	1	1
	Losses	5	1	2	2	0	3	1	5	5
	Difference	-4	+4	+1	+2	+6	0	+3	-4	-4
	Rank	6	2	3	3	1	4	2	6	6
MOEA/D	Wins	5	1	1	2	3	4	1	4	3
	Losses	0	5	5	4	3	2	5	1	3
	Difference	+5	-4	-4	-2	0	+2	-4	+3	0
	Rank	1	6	6	5	4	3	6	2	4
KnDE	Wins	3	0	0	0	1	2	0	0	4
	Losses	3	6	6	6	5	4	6	6	2
	Difference	0	-6	-6	-6	-4	-2	-6	-6	+2
	Rank	4	7	7	7	6	5	7	7	3
KnPSO	Wins	5	6	5	5	5	5	3	3	6
	Losses	0	0	1	1	1	0	3	3	0
	Difference	+5	+6	+4	+4	+4	+5	0	0	+6
	Rank	1	1	2	2	2	1	4	4	1
SrEA/D	Wins	0	4	3	3	2	5	4	4	2
	Losses	6	2	2	3	4	0	1	1	4
	Difference	-6	+2	+1	0	-2	+5	+3	+3	-2
	Rank	7	3	3	4	5	1	2	2	5
CDAS-SMPSO	Wins	4	3	6	6	4	0	6	6	5
	Losses	2	3	0	0	2	6	0	0	1
	Difference	+2	0	+6	+6	+2	-6	+6	+6	+4
	Rank	3	4	1	1	3	7	1	1	2
KnEA	Wins	2	2	2	1	0	1	2	2	0
	Losses	4	4	4	5	6	5	4	4	6
	Difference	-2	-2	-2	-4	-6	-4	-2	-2	-6
	Rank	5	5	5	6	7	6	5	5	7

Table 9.16: IGD Rank Summary Using 1000 Decision Variables, 10 Objectives

	Algorithm						
Measure	NSGA-III	MOEA/D	KnDE	KnPSO	SrEA/D	CDAS-SMPSO	KnEA
Mean	3.667	4.111	5.889	**2.0**	3.556	2.556	5.667
Standard Deviation	1.826	1.728	1.449	1.155	1.771	1.892	0.816
Maximum	6.0	6.0	7.0	4.0	7.0	7.0	7.0
Minimum	1.0	1.0	3.0	1.0	1.0	1.0	5.0

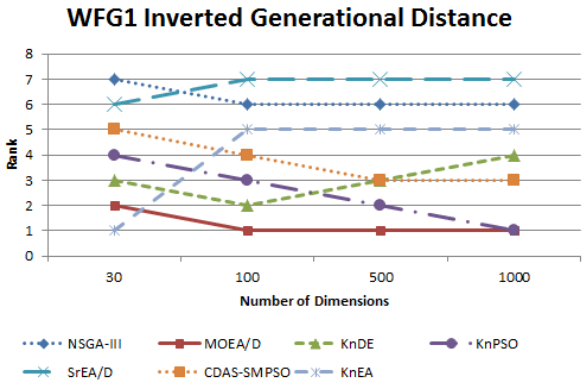

Figure 9.10: Ranking of IGD metric vs. number of decision variables for the WFG1 problem

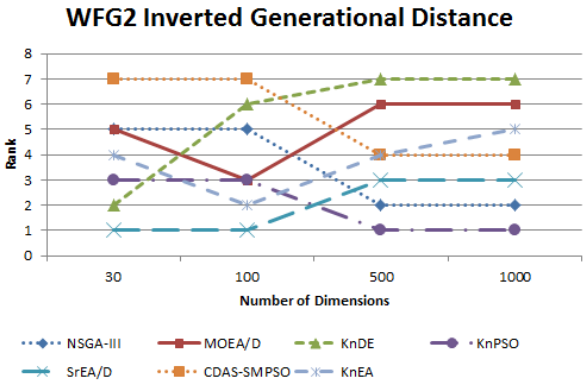

Figure 9.11: Ranking of IGD metric vs. number of decision variables for the WFG2 problem

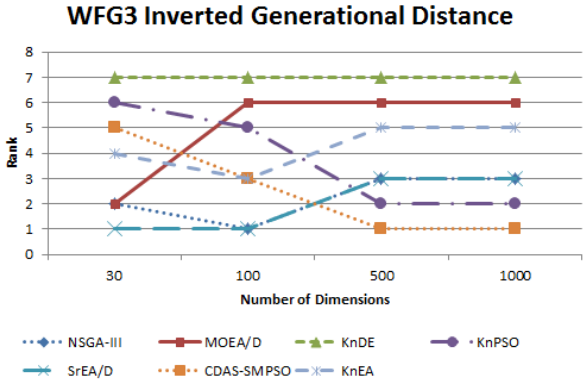

Figure 9.12: Ranking of IGD metric vs. number of decision variables for the WFG3 problem

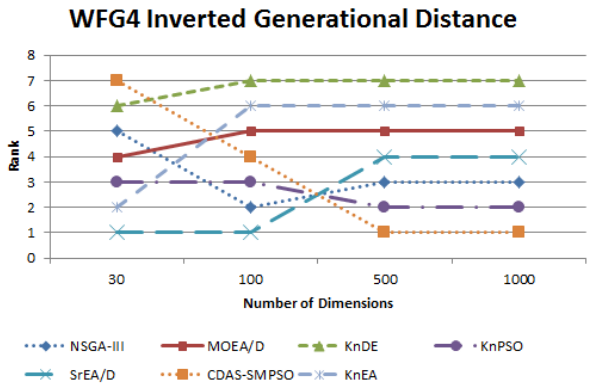

Figure 9.13: Ranking of IGD metric vs. number of decision variables for the WFG4 problem

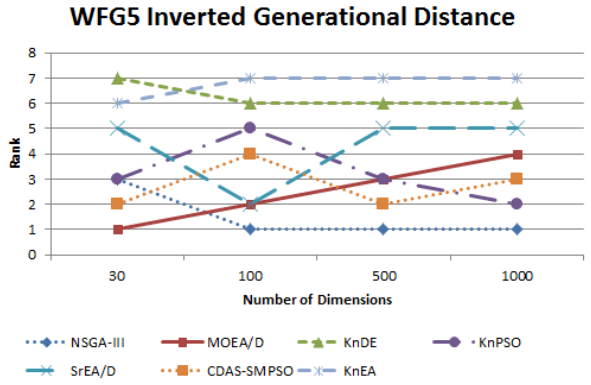

Figure 9.14: Ranking of IGD metric vs. number of decision variables for the WFG5 problem

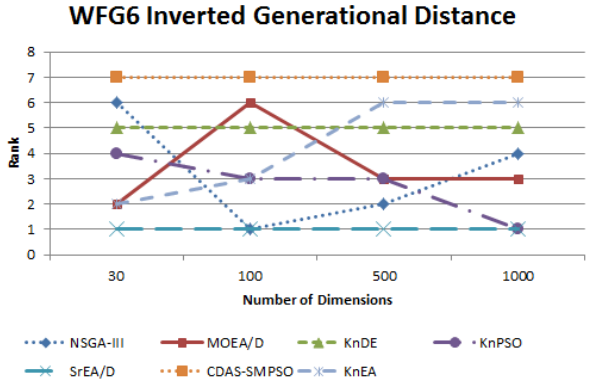

Figure 9.15: Ranking of IGD metric vs. number of decision variables for the WFG6 problem

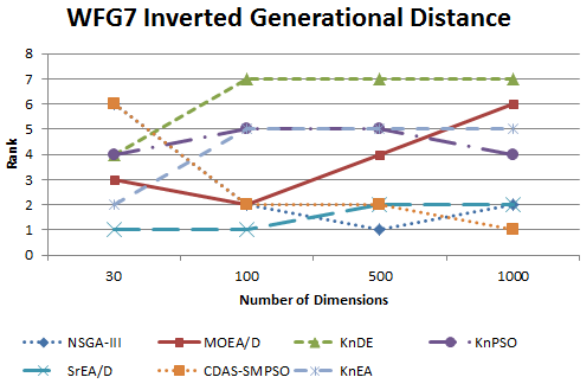

Figure 9.16: Ranking of IGD metric vs. number of decision variables for the WFG7 problem

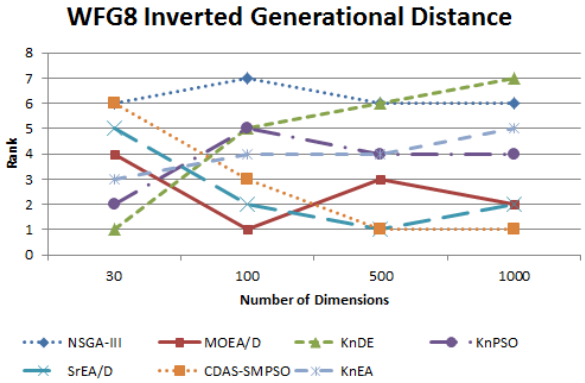

Figure 9.17: Ranking of IGD metric vs. number of decision variables for the WFG8 problem

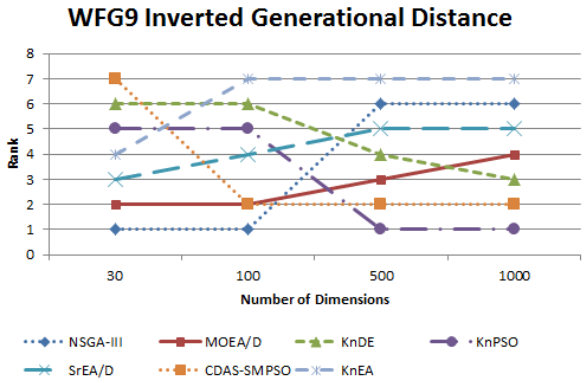

Figure 9.18: Ranking of IGD metric vs. number of decision variables for the WFG9 problem

Chapter 10

Concluding Remarks and Future Work

This thesis explored the challenges of many-objective optimization, proposing novel algorithms designed to solve MaOPs. Two algorithms incorporating the concept of knee points were proposed, inspired by concepts from an existing knee-driven evolutionary algorithm in [108]. Additionally, a hybrid evolutionary algorithm was proposed, which utilizes a sum-of-ranks [6] technique to converge while concurrently maintaining a well-spread set of solutions using a decomposition-based approach. Experiments directly comparing each knee-driven algorithm to their non-knee counterparts were performed to determine the exact performance gain experienced via the incorporation of knee points. Additionally, each of the three algorithms proposed in this thesis were compared with several state-of-the-art many-objective optimizers which incorporate a variety of diverse techniques described in Section 4.2. All algorithms were compared over an increasing number of objectives and increasing number of decision variables to determine the scalability of each optimizer. A variety of MOPs were used for benchmarking purposes, each incorporating a unique set of difficulties commonly seen in practical problems.

Results indicated the incorporation of knee points to improve both the hypervolume and IGD metric values produced by Pareto-optimizers. Utilizing knee points

seemed to improve the selection pressure to converge to the true Pareto-optimal front, while also maintaining a good spread of solutions. It was noted that functions with degenerate linear landscapes are the exception to this, as all knee point algorithms experienced significant performance degradation in its presence. However, when the number of decision variables were increased, degenerate linear landscapes presented less of a problem for the knee-driven optimizers.

Several notable conclusions were drawn from the objective scalability experiments comparing the proposed algorithms to state-of-the-art optimizers. The SrEA/D algorithm was shown to perform promisingly, often obtaining better IGD and hypervolume values than the other tested optimizers. The combination of the NF_{sum} method along with the utilization of reference points to maintain a near-uniform spread of solutions often produced a diverse, well-converged set of solutions. Other approaches which performed well for MaOPs possessing a low number of decision variables (30) was the decomposition-based strategy of MOEA/D and the knee-driven approach of KnEA. Both algorithms achieved a good blend of both convergence and diversity, producing a set of well-distributed converged solutions.

Observations seen for the decision variable scalability experiments were considerably different than those of the objective scalability experiments. The proposed KnPSO algorithm performed very well overall, possessing the best hypervolume ranking and the second best IGD metric rank on average out of all algorithms. CDAS-SMPSO also performed notably well, experiencing drastically improved hypervolume and IGD values on MaOPs when the number of decision variables is increased. These observations suggest that evolutionary approaches may perform worse than PSO approaches for MaOPs with large numbers of decision variables, likely due to the crossover operator producing children distant from both parents as a result of the immense search space size. This phenomena may play a role in the poor performance of KnEA and KnDE on large-scale MaOPs. Considerable empirical evidence within this thesis

also supports the use of decomposition-based strategies and sum-of-rank methods for solving MaOPs possessing large amounts of decision variables (500-1000+). Algorithms implementing these methods performed competitively, yielding consistent performance for the majority of functions.

There are many intriguing avenues for future work in this area. This thesis has demonstrated that a number of solutions currently exist for many-objective optimization, however the issue as a whole is still far from solved. It is important to propose creative new algorithms with the intent of further addressing the unique issues seen for many-objective optimization. The current state of the literature is such that most many-objective optimizers obtain either Pareto-optimal front convergence or a well-spread set of solutions, but not both simultaneously. Investigating possible remedies to scale many-objective optimizers to large-scale MaOPs is another interesting offshoot of this thesis. Considerable empirical evidence within this work supports the conclusion that some many-objective optimizers degrade when presented with an increasing number of decision variables. Addressing the crossover problem present for evolutionary many-objective optimizers is an important future work avenue, however one should note that a mating restriction scheme was shown to remedy the overall issue to a degree in this work. With regards to the knee-driven algorithms proposed in this work, the knee identification mechanism can be further improved to be more computationally efficient and effective overall.

Bibliography

[1] M. Ali and A. Törn. Population set-based global optimization algorithms: Some modifications and numerical studies. *Journal on Computers and Operations Research*, 31(10):1703–1725, September 2004.

[2] J. Bader, K. Deb, and E. Zitzler. Faster hypervolume-based search using monte carlo sampling. In *Multiple Criteria Decision Making for Sustainable Energy and Transportation Systems*, volume 634 of *Lecture Notes in Economics and Mathematical Systems*, pages 313–326. Springer Berlin Heidelberg, 2010.

[3] J. Bader and E. Zitzler. Hype: An algorithm for fast hypervolume-based many-objective optimization. *Evolutionary Computation*, 19(1):45–76, March 2011.

[4] W. Barry and B. Ross. Virtual photography using multi-objective particle swarm optimization. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, GECCO '14, pages 285–292, New York, NY, USA, 2014. ACM.

[5] D. Beasley, D. Bull, and R. Martin. An overview of genetic algorithms: Part 1, fundamentals. *University Computing*, 15(2):58–69, 1993.

[6] P.J. Bentley and J.P. Wakefield. Finding acceptable solutions in the pareto-optimal range using multiobjective genetic algorithms. In *Soft Computing in Engineering Design and Manufacturing*, pages 231–240. Springer London, 1998.

[7] N. Beume, B. Naujoks, and M. Emmerich. Sms-emoa: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3):1653 – 1669, 2007.

[8] L. Bradstreet, University of Western Australia. School of Computer Science, and Software Engineering. *The Hypervolume Indicator for Multi-objective Optimisation: Calculation and Use*. University of Western Australia, 2011.

[9] J. Brest, V. Zumer, and M.S. Maucec. Self-adaptive differential evolution algorithm in constrained real-parameter optimization. In *IEEE International Congress on Evolutionary Computation*, pages 215–222, 2006.

[10] A. Brindle. *Genetic algorithms for function optimization*. PhD thesis, Edmonton, Calgary, CA, 1980.

[11] K. Bringmann and T. Friedrich. An efficient algorithm for computing hypervolume contributions. *Evolutionary Computation*, page 2010.

[12] K. Bringmann and T. Friedrich. Approximating the volume of unions and intersections of high-dimensional geometric objects. In Seok-Hee Hong, Hiroshi Nagamochi, and Takuro Fukunaga, editors, *Algorithms and Computation*, volume 5369 of *Lecture Notes in Computer Science*, pages 436–447. Springer Berlin Heidelberg, 2008.

[13] A. Carlisle. *Applying the Particle Swarm Optimizer to Non-stationary Environments*. PhD thesis, Auburn, AL, USA, 2002. AAI3070763.

[14] A. Carlisle and G. Dozier. An Off-The-Shelf PSO. 2001.

[15] R. Chelouah and C. Baron. Ant colony algorithm hybridized with tabu and greedy searches as applied to multi-objective optimization in project management. *Journal of Heuristics*, 13(6):640–640, 2007.

[16] C. Chow and H. Tsui. Autonomous agent response learning by a multi-species particle swarm optimization. In *IEEE International Congress on Evolutionary Computation*, volume 1, pages 778–785 Vol.1, June 2004.

[17] M. Clerc. The swarm and the queen: towards a deterministic and adaptive particle swarm optimization. In *Proceedings of the 1999 Congress on Evolutionary Computation*, volume 3, pages –1957 Vol. 3, 1999.

[18] D. Corne and J. Knowles. Techniques for highly multiobjective optimisation: Some nondominated points are better than others. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*, GECCO '07, pages 773–780, New York, NY, USA, 2007. ACM.

[19] R. Crump, V. Hotz, G. Imbens, and O. Mitnik. Nonparametric tests for treatment effect heterogeneity. Working Paper 324, National Bureau of Economic Research, June 2006.

[20] I. Das. A preference ordering among various pareto optimal alternatives. *Structural optimization*, 18(1):30–35, 1999.

[21] I. Das and J. E. Dennis. Normal-boundary intersection: A new method for generating the pareto surface in nonlinear multicriteria optimization problems. *SIAM Journal on Optimization*, 8(3):631–657, 1998.

[22] A. B. de Carvalho and A. Pozo. Measuring the convergence and diversity of cdas multi-objective particle swarm optimization algorithms: A study of many-objective problems. *Neurocomputing*, 75(1):43 – 51, 2012. Brazilian Symposium on Neural Networks: International Conference on Hybrid Artificial Intelligence Systems.

[23] K. A. De Jong. *An Analysis of the Behavior of a Class of Genetic Adaptive Systems.* PhD thesis, Ann Arbor, MI, USA, 1975. AAI7609381.

[24] K. Deb. *Multi-Objective Optimization Using Evolutionary Algorithms.* John Wiley & Sons, Inc., New York, NY, USA, 2001.

[25] K. Deb and R. B. Agrawal. Simulated Binary Crossover for Continuous Search Space. 1995.

[26] K. Deb and M. Goyal. A combined genetic adaptive search for engineering design. *Computer Science and Informatics*, 26:30–45, 1996.

[27] K. Deb and H. Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints. *IEEE Transactions on Evolutionary Computation*, 18(4):577–601, Aug 2014.

[28] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, Apr 2002.

[29] F. di Pierro, S Khu, and D.A. Savic. An investigation on preference order ranking scheme for multiobjective evolutionary optimization. *IEEE Transactions on Evolutionary Computation*, 11(1):17–45, Feb 2007.

[30] F. Djeffal and N. Lakhdar. An improved analog electrical performance of submicron dual-material gate (dm) gaas-mesfets using multi-objective computation. *Journal of Computational Electronics*, 12(1):29–35, 2013.

[31] J. J. Durillo and A. J. Nebro. jmetal: A java framework for multi-objective optimization. *Advances in Engineering Software*, 42(10):760 – 771, 2011.

[32] R. Eberhart, P. Simpson, and R. Dobbins. *Computational Intelligence PC Tools*. Academic Press Professional, Inc., San Diego, CA, USA, 1996.

[33] A. P. Engelbrecht. Particle swarm optimization: Velocity initialization. In *IEEE International Congress on Evolutionary Computation*, pages 1–8, June 2012.

[34] A. P. Engelbrecht. Fitness function evaluations: A fair stopping condition? In *IEEE Symposium on Swarm Intelligence*, pages 1–8, Dec 2014.

[35] M. Fleischer. The measure of pareto optima applications to multi-objective metaheuristics. In *Proceedings of the 2Nd International Conference on Evolutionary Multi-criterion Optimization*, EMO'03, pages 519–533, Berlin, Heidelberg, 2003. Springer-Verlag.

[36] G. Fu, Z. Kapelan, J. Kasprzyk, and P. Reed. Optimal design of water distribution systems using many-objective visual analytics. *Journal of Water Resources Planning and Management*, 139(6):624–633, 2013.

[37] M. Garza-Fabre, G. T. Pulido, and C. A. Coello Coello. *Proceedings of the 8th Mexican International Conference on Artificial Intelligence*, chapter Ranking Methods for Many-Objective Optimization, pages 633–645. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[38] M. Gen, Y. Tsujimura, and E. Kubota. Solving job-shop scheduling problems by genetic algorithm. In *IEEE International Conference on Systems, Man, and Cybernetic*, volume 2, pages 1577–1582 vol.2, Oct 1994.

[39] C.K. Goh and K.C. Tan. An investigation on noisy environments in evolutionary multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 11(3):354–381, June 2007.

[40] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1989.

[41] D. E. Goldberg and K. Deb. A comparative analysis of selection schemes used in genetic algorithms. In *Foundations of Genetic Algorithms*, pages 69–93. Morgan Kaufmann, 1991.

[42] N. Hansen and A. Ostermeier. Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation. In *Proceedings of IEEE International Conference on Evolutionary Computation*, pages 312–317, May 1996.

[43] K.R. Harrison, AP. Engelbrecht, and B.M. Ombuki-Berman. A scalability study of multi-objective particle swarm optimizers. In *IEEE Congress on Evolutionary Computation*, pages 189–197, June 2013.

[44] Z. He, G.G. Yen, and G. Zhang. Fuzzy-based pareto optimality for many-objective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 18(2):269–285, April 2014.

[45] J.S. Heo, K.Y. Lee, and R. Garduno-Ramirez. Multiobjective control of power plants using particle swarm optimization techniques. *IEEE Transactions on Energy Conversion*, 21(2):552–561, June 2006.

[46] S.L. Ho, S. Yang, G. Ni, and H. C. Wong. A particle swarm optimization method with enhanced global search ability for design optimizations of electromagnetic devices. *IEEE Transactions on Magnetics*, 42(4):1107–1110, April 2006.

[47] J. H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press, Cambridge, MA, USA, 1992.

[48] X. Hu, R.C. Eberhart, and Y. Shi. Swarm intelligence for permutation optimization: a case study of n-queens problem. In *Proceedings of the 2003 IEEE Swarm Intelligence Symposium*, pages 243–246, April 2003.

[49] S. Huband, L. Barone, L. While, and P. Hingston. A scalable multi-objective test problem toolkit. In *Evolutionary Multi-Criterion Optimization*, volume 3410 of *Lecture Notes in Computer Science*, pages 280–295. Springer Berlin Heidelberg, 2005.

[50] S. Huband, P. Hingston, L. Barone, and L. While. A review of multiobjective test problems and a scalable test problem toolkit. *IEEE Transactions on Evolutionary Computation*, 10(5):477–506, Oct 2006.

[51] C. Igel, N. Hansen, and S. Roth. Covariance matrix adaptation for multi-objective optimization. *Evolutionary Computation*, 15(1):1–28, March 2007.

[52] K. Ikeda, H. Kita, and S. Kobayashi. Failure of pareto-based moeas: does non-dominated really mean near to optimal? In *Proceedings of IEEE International Congress on Evolutionary Computation*, volume 2, pages 957–962 vol. 2, 2001.

[53] H. Ishibuchi, N. Tsukamoto, and Y. Nojima. Evolutionary many-objective optimization: A short review. In *IEEE International Congress on Evolutionary Computation*, pages 2419–2426, June 2008.

[54] I.N. Kassabalidis, M.A. El-Sharkawi, R.J. Marks, L.S. Moulin, and A.P. Alves da Silva. Dynamic security border identification using enhanced particle swarm optimization. *Power Systems, IEEE Transactions on*, 17(3):723–729, Aug 2002.

[55] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of the IEEE International Conference on Neural Networks*, volume 4, pages 1942–1948, Nov 1995.

[56] S. Kitayama, J. Srirat, M. Arakawa, and K. Yamazaki. Sequential approximate multi-objective optimization using radial basis function network. *Structural and Multidisciplinary Optimization*, 48(3):501–515, 2013.

[57] J. Knowles, L. Thiele, and E. Zitzler. A Tutorial on the Performance Assessment of Stochastic Multiobjective Optimizers. Technical report, Computer Engineering and Networks Laboratory (TIK), ETH Zurich, Switzerland, 2006.

[58] M. Koppen and K. Yoshida. Substitute distance assignments in nsga-ii for handling many-objective optimization problems. In *Evolutionary Multi-Criterion Optimization*, volume 4403 of *Lecture Notes in Computer Science*, pages 727–741. Springer Berlin Heidelberg, 2007.

[59] S. Kukkonen and K. Deb. Improved pruning of non-dominated solutions based on crowding distance for bi-objective optimization problems. In *IEEE Congress on Evolutionary Computation*, pages 1179–1186, 2006.

[60] S. Kukkonen and J. Lampinen. Comparison of generalized differential evolution algorithm to other multi-objective evolutionary algorithms. In *Proceedings of*

the 4th European Congress on Computational Methods in Applied Sciences and Engineering (ECCO- MAS2004), page 445, jul 2004.

[61] S. Kukkonen and J. Lampinen. An extension of generalized differential evolution for multi-objective optimization with constraints. In *Parallel Problem Solving from Nature - PPSN VIII*, volume 3242 of *Lecture Notes in Computer Science*, pages 752–761. Springer Berlin Heidelberg, 2004.

[62] S. Kukkonen and J. Lampinen. Gde3: the third evolution step of generalized differential evolution. In *IEEE International Congress on Evolutionary Computation*, volume 1, pages 443–450 Vol.1, Sept 2005.

[63] S. Kukkonen and J. Lampinen. Ranking-dominance and many-objective optimization. In *IEEE International Congress on Evolutionary Computation*, pages 3983–3990, Sept 2007.

[64] M. Laumanns, L. Thiele, K. Deb, and E. Zitzler. Combining convergence and diversity in evolutionary multiobjective optimization. *Evolutionary Computation*, 10(3):263–282, September 2002.

[65] K. Li, K. Deb, Q. Zhang, and S. Kwong. An evolutionary many-objective optimization algorithm based on dominance and decomposition. *IEEE Transactions on Evolutionary Computation*, 19(5):694–716, Oct 2015.

[66] M. Li, J. Zheng, R. Shen, K. Li, and Q. Yuan. A grid-based fitness strategy for evolutionary many-objective optimization. In *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*, GECCO '10, pages 463–470, New York, NY, USA, 2010. ACM.

[67] X. Li. *Genetic and Evolutionary Computation — GECCO 2003: Genetic and Evolutionary Computation Conference Chicago, IL, USA, July 12–16, 2003 Proceedings, Part I*, chapter A Non-dominated Sorting Particle Swarm Optimizer for Multiobjective Optimization, pages 37–48. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

[68] Y. Li, K. C. Ng, D. J. Murray-Smith, G. J. Gray, and K. C. Sharman. Genetic algorithm automated approach to design of sliding mode control systems. *International Journal of Control*, 63:721–739, 1996.

[69] H. Liu, F. Gu, and Q. Zhang. Decomposition of a multiobjective optimization problem into a number of simple multiobjective subproblems. *IEEE Transactions on Evolutionary Computation*, 18(3):450–455, June 2014.

[70] W. Liu. Design of a multiband cpw-fed monopole antenna using a particle swarm optimization approach. *IEEE Transactions on Antennas and Propagation*, 53(10):3273–3279, Oct 2005.

[71] R. J. Lygoe, M. Cary, and P. J. Fleming. A real-world application of a many-objective optimisation complexity reduction process. In *Evolutionary Multi-Criterion Optimization*, volume 7811 of *Lecture Notes in Computer Science*, pages 641–655. Springer Berlin Heidelberg, 2013.

[72] H. B. Mann and D. R. Whitney. On a test of whether one of two random variables is stochastically larger than the other. *The Annals of Mathematical Statistics*, 18(1):50–60, 03 1947.

[73] M. Matsumoto and T. Nishimura. Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation*, 8(1):3–30, January 1998.

[74] J.M. Mendel. Fuzzy logic systems for engineering: a tutorial. *Proceedings of the IEEE*, 83(3):345–377, Mar 1995.

[75] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs (3rd Edition)*. Springer-Verlag, London, UK, 1996.

[76] M. Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA, USA, 1998.

[77] K. Narukawa and T. Rodemann. Examining the performance of evolutionary many-objective optimization algorithms on a real-world application. In *Sixth International Conference on Genetic and Evolutionary Computing*, pages 316–319, Aug 2012.

[78] AJ. Nebro, J.J. Durillo, J. Garcia-Nieto, C.A Coello Coello, F. Luna, and E. Alba. Smpso: A new pso-based metaheuristic for multi-objective optimization. In *IEEE Symposium on Computational Intelligence in Multi-Criteria Decision-Making*, pages 66–73, March 2009.

[79] I. M. Oliver, D. J. Smith, and J. R. C. Holland. A study of permutation crossover operators on the traveling salesman problem. In *Proceedings of the Second International Conference on Genetic Algorithms and Their Applications*, pages 224–230, Hillsdale, NJ, USA, 1987. L. Erlbaum Associates Inc.

[80] B. M. Ombuki-Berman, A. Runka, and F T. Hanshar. Waste collection vehicle routing problem with time windows using multi-objective genetic algorithms. In *Proceedings of the Third IASTED International Conference on Computational Intelligence*, pages 91–97, Anaheim, CA, USA, 2007. ACTA Press.

[81] M. Omran, A. Salman, and A. P. Engelbrecht. Self-adaptive differential evolution. In *Computational Intelligence and Security*, volume 3801 of *Lecture Notes in Computer Science*, pages 192–199. Springer Berlin Heidelberg, 2005.

[82] K. E. Parsopoulos, D. K. Tasoulis, and M. N. Vrahatis. Multiobjective optimization using parallel vector evaluated particle swarm optimization. In *Proceedings of the IASTED International Conference on Artificial Intelligence and Applications*, pages 823–828. ACTA Press, 2004.

[83] K. E. Parsopoulos and M. N. Vrahatis. Particle swarm optimization method in multiobjective problems. In *Proceedings of the 2002 ACM Symposium on Applied Computing*, SAC '02, pages 603–607, New York, NY, USA, 2002. ACM.

[84] K. V. Price. New ideas in optimization. chapter An Introduction to Differential Evolution, pages 79–108. McGraw-Hill Ltd., UK, Maidenhead, UK, England, 1999.

[85] A.K. Qin and P.N. Suganthan. Self-adaptive differential evolution algorithm for numerical optimization. In *IEEE International Congress on Evolutionary Computation*, volume 2, pages 1785–1791 Vol. 2, Sept 2005.

[86] C. Reeves. Hybrid genetic algorithms for bin-packing and related problems. *Annals of Operations Research*, 63(3):371–396.

[87] M. Rosendo and A. Pozo. Applying a discrete particle swarm optimization algorithm to combinatorial problems. In *Eleventh Brazilian Symposium on Neural Networks*, pages 235–240, Oct 2010.

[88] B.J. Ross. Evolution of stochastic bio-networks using summed rank strategies. In *IEEE International Congress on Evolutionary Computation*, pages 773–780, June 2011.

[89] H. Sato, H. Aguirre, and K. Tanaka. Self-controlling dominance area of solutions in evolutionary many-objective optimization. In *Simulated Evolution and Learning*, volume 6457 of *Lecture Notes in Computer Science*, pages 455–465. Springer Berlin Heidelberg, 2010.

[90] H. Sato, H. E. Aguirre, and K. Tanaka. Controlling dominance area of solutions and its impact on the performance of moeas. In *Proceedings of the 4th International Conference on Evolutionary Multi-criterion Optimization*, EMO'07, pages 5–20, Berlin, Heidelberg, 2007. Springer-Verlag.

[91] D. K. Saxena, J. Duro, A. Tiwari, K. Deb, and Q. Zhang. Objective reduction in many-objective optimization: Linear and nonlinear algorithms. *IEEE Transactions on Evolutionary Computation*, 17(1):77–99, February 2013.

[92] O. Schutze, A. Lara, and Carlos A. Coello Coello. On the influence of the number of objectives on the hardness of a multiobjective optimization problem. *IEEE Transactions on Evolutionary Computation*, 15(4):444–455, Aug 2011.

[93] X.H. Shi, Y. Zhou, L.M. Wang, Q.X. Wang, and Y.C. Liang. A discrete particle swarm optimization algorithm for travelling salesman problem. In *Computational Methods*, pages 1063–1068. Springer Netherlands, 2006.

[94] Y. Shi and R. C. Eberhart. Parameter selection in particle swarm optimization. In *Evolutionary Programming VII*, volume 1447 of *Lecture Notes in Computer Science*, pages 591–600. Springer Berlin Heidelberg, 1998.

[95] M. R. Sierra and C. A. Coello Coello. Improving pso-based multi-objective optimization using crowding, mutation and dominance. In *Evolutionary Multi-objective Optimization*, pages 505–519. Springer-Verlag, 2005.

[96] N. Srinivas and K. Deb. Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3):221–248, September 1994.

[97] J. E. Stiglitz. Self-selection and Pareto efficient taxation. *Journal of Public Economics*, 17(2):213–240, March 1982.

[98] R. Storn and K. Price. Differential evolution: A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359, December 1997.

[99] A. Sülflow, N. Drechsler, and R. Drechsler. *Evolutionary Multi-Criterion Optimization: 4th International Conference, EMO 2007, Matsushima, Japan, March 5-8, 2007. Proceedings*, chapter Robust Multi-Objective Optimization in High Dimensional Spaces, pages 715–726. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[100] G.S. Tewolde, D.M. Hanna, and R.E. Haskell. Enhancing performance of pso with automatic parameter tuning technique. In *IEEE Symposium on Swarm Intelligence*, pages 67–73, March 2009.

[101] F. Van Den Bergh. *An Analysis of Particle Swarm Optimizers*. PhD thesis, Pretoria, South Africa, South Africa, 2002. AAI0804353.

[102] L. While, L. Bradstreet, and L. Barone. A fast way of calculating exact hypervolumes. *IEEE Transactions on Evolutionary Computation*, 16(1):86–95, Feb 2012.

[103] L. While, P. Hingston, L. Barone, and S. Huband. A faster algorithm for calculating hypervolume. *IEEE Transactions on Evolutionary Computation*, 10(1):29–38, Feb 2006.

[104] S. Yang, M. Li, X. Liu, and J. Zheng. A grid-based evolutionary algorithm for many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 17(5):721–736, Oct 2013.

[105] Y. Yuan and H. Xu. Multiobjective flexible job shop scheduling using memetic algorithms. *IEEE Transactions on Automation Science and Engineering*, 12(1):336–353, Jan 2015.

[106] Q. Zhang and H. Li. Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731, Dec 2007.

[107] X. Zhang, Y. Tian, R. Cheng, and Y. Jin. An efficient approach to nondominated sorting for evolutionary multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 19(2):201–213, April 2015.

[108] X. Zhang, Y. Tian, and Y. Jin. A knee point driven evolutionary algorithm for many-objective optimization. *IEEE Transactions on Evolutionary Computation*, PP(99):1–1, 2014.

[109] W. Zhao, C. Wang, L. Yu, and T. Chen. Performance optimization of electric power steering based on multi-objective genetic algorithm. *Journal of Central South University*, 20(1):98–104, 2013.

[110] E. Zitzler. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications.* PhD thesis, ETH Zurich, 1999.

[111] E. Zitzler, K. Deb, and L. Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8(2):173–195, June 2000.

[112] E. Zitzler and S. Kunzli. Indicator-based selection in multiobjective search. In *Parallel Problem Solving from Nature - PPSN VIII*, volume 3242 of *Lecture Notes in Computer Science*, pages 832–842. Springer Berlin Heidelberg, 2004.

[113] E. Zitzler and L. Thiele. Multiobjective optimization using evolutionary algorithms: A comparative case study. In *Parallel Problem Solving from Nature PPSN V*, volume 1498 of *Lecture Notes in Computer Science*, pages 292–301. Springer Berlin Heidelberg, 1998.

[114] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, Nov 1999.

[115] E. Zitzler, L. Thiele, M. Laumanns, C.M. Fonseca, and V.G. da Fonseca. Performance assessment of multiobjective optimizers: an analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132, April 2003.

[116] X. Zou, Y Chen, M. Liu, and L. Kang. A new evolutionary algorithm for solving many-objective optimization problems. *IEEE Transactions on Systems, Man, and Cybernetics*, 38(5):1402–1412, Oct 2008.

Appendix A

Summary of Experiments Performed

Table A.1 presents a comprehensive summary of all experiments performed within this work. Each experiment was repeated 30 times to ensure statistical significance.

Table A.1: Experiments Performed

Algorithm	**Objectives**	**Decision Variables**	**Function**
NSGA-II	2	30	WFG1
NSGA-II	2	30	WFG2
NSGA-II	2	30	WFG3
NSGA-II	2	30	WFG4
NSGA-II	2	30	WFG5
NSGA-II	2	30	WFG6
NSGA-II	2	30	WFG7
NSGA-II	2	30	WFG8

Continued on next page

Table A.1: Experiments Performed (continued)

Algorithm	**Objectives**	**Decision Variables**	**Function**
NSGA-II	2	30	WFG9
NSGA-II	4	30	WFG1
NSGA-II	4	30	WFG2
NSGA-II	4	30	WFG3
NSGA-II	4	30	WFG4
NSGA-II	4	30	WFG5
NSGA-II	4	30	WFG6
NSGA-II	4	30	WFG7
NSGA-II	4	30	WFG8
NSGA-II	4	30	WFG9
NSGA-II	6	30	WFG1
NSGA-II	6	30	WFG2
NSGA-II	6	30	WFG3
NSGA-II	6	30	WFG4
NSGA-II	6	30	WFG5
NSGA-II	6	30	WFG6
NSGA-II	6	30	WFG7

Continued on next page

Table A.1: Experiments Performed (continued)

Algorithm	**Objectives**	**Decision Variables**	**Function**
NSGA-II	6	30	WFG8
NSGA-II	6	30	WFG9
NSGA-II	8	30	WFG1
NSGA-II	8	30	WFG2
NSGA-II	8	30	WFG3
NSGA-II	8	30	WFG4
NSGA-II	8	30	WFG5
NSGA-II	8	30	WFG6
NSGA-II	8	30	WFG7
NSGA-II	8	30	WFG8
NSGA-II	8	30	WFG9
NSGA-II	10	30	WFG1
NSGA-II	10	30	WFG2
NSGA-II	10	30	WFG3
NSGA-II	10	30	WFG4
NSGA-II	10	30	WFG5
NSGA-II	10	30	WFG6

Continued on next page

Table A.1: Experiments Performed (continued)

Algorithm	**Objectives**	**Decision Variables**	**Function**
NSGA-II	10	30	WFG7
NSGA-II	10	30	WFG8
NSGA-II	10	30	WFG9
GDE3	2	30	WFG1
GDE3	2	30	WFG2
GDE3	2	30	WFG3
GDE3	2	30	WFG4
GDE3	2	30	WFG5
GDE3	2	30	WFG6
GDE3	2	30	WFG7
GDE3	2	30	WFG8
GDE3	2	30	WFG9
GDE3	4	30	WFG1
GDE3	4	30	WFG2
GDE3	4	30	WFG3
GDE3	4	30	WFG4
GDE3	4	30	WFG5

Continued on next page

Table A.1: Experiments Performed (continued)

Algorithm	**Objectives**	**Decision Variables**	**Function**
GDE3	4	30	WFG6
GDE3	4	30	WFG7
GDE3	4	30	WFG8
GDE3	4	30	WFG9
GDE3	6	30	WFG1
GDE3	6	30	WFG2
GDE3	6	30	WFG3
GDE3	6	30	WFG4
GDE3	6	30	WFG5
GDE3	6	30	WFG6
GDE3	6	30	WFG7
GDE3	6	30	WFG8
GDE3	6	30	WFG9
GDE3	8	30	WFG1
GDE3	8	30	WFG2
GDE3	8	30	WFG3
GDE3	8	30	WFG4

Continued on next page

Table A.1: Experiments Performed (continued)

Algorithm	**Objectives**	**Decision Variables**	**Function**
GDE3	8	30	WFG5
GDE3	8	30	WFG6
GDE3	8	30	WFG7
GDE3	8	30	WFG8
GDE3	8	30	WFG9
GDE3	10	30	WFG1
GDE3	10	30	WFG2
GDE3	10	30	WFG3
GDE3	10	30	WFG4
GDE3	10	30	WFG5
GDE3	10	30	WFG6
GDE3	10	30	WFG7
GDE3	10	30	WFG8
GDE3	10	30	WFG9
SMPSO	2	30	WFG1
SMPSO	2	30	WFG2
SMPSO	2	30	WFG3

Continued on next page

Table A.1: Experiments Performed (continued)

Algorithm	**Objectives**	**Decision Variables**	**Function**
SMPSO	2	30	WFG4
SMPSO	2	30	WFG5
SMPSO	2	30	WFG6
SMPSO	2	30	WFG7
SMPSO	2	30	WFG8
SMPSO	2	30	WFG9
SMPSO	4	30	WFG1
SMPSO	4	30	WFG2
SMPSO	4	30	WFG3
SMPSO	4	30	WFG4
SMPSO	4	30	WFG5
SMPSO	4	30	WFG6
SMPSO	4	30	WFG7
SMPSO	4	30	WFG8
SMPSO	4	30	WFG9
SMPSO	6	30	WFG1
SMPSO	6	30	WFG2

Continued on next page

Table A.1: Experiments Performed (continued)

Algorithm	**Objectives**	**Decision Variables**	**Function**
SMPSO	6	30	WFG3
SMPSO	6	30	WFG4
SMPSO	6	30	WFG5
SMPSO	6	30	WFG6
SMPSO	6	30	WFG7
SMPSO	6	30	WFG8
SMPSO	6	30	WFG9
SMPSO	8	30	WFG1
SMPSO	8	30	WFG2
SMPSO	8	30	WFG3
SMPSO	8	30	WFG4
SMPSO	8	30	WFG5
SMPSO	8	30	WFG6
SMPSO	8	30	WFG7
SMPSO	8	30	WFG8
SMPSO	8	30	WFG9
SMPSO	10	30	WFG1

Continued on next page

Table A.1: Experiments Performed (continued)

Algorithm	**Objectives**	**Decision Variables**	**Function**
SMPSO	10	30	WFG2
SMPSO	10	30	WFG3
SMPSO	10	30	WFG4
SMPSO	10	30	WFG5
SMPSO	10	30	WFG6
SMPSO	10	30	WFG7
SMPSO	10	30	WFG8
SMPSO	10	30	WFG9
CDAS-SMPSO	2	30	WFG1
CDAS-SMPSO	2	30	WFG2
CDAS-SMPSO	2	30	WFG3
CDAS-SMPSO	2	30	WFG4
CDAS-SMPSO	2	30	WFG5
CDAS-SMPSO	2	30	WFG6
CDAS-SMPSO	2	30	WFG7
CDAS-SMPSO	2	30	WFG8
CDAS-SMPSO	2	30	WFG9

Continued on next page

Table A.1: Experiments Performed (continued)

Algorithm	**Objectives**	**Decision Variables**	**Function**
CDAS-SMPSO	4	30	WFG1
CDAS-SMPSO	4	30	WFG2
CDAS-SMPSO	4	30	WFG3
CDAS-SMPSO	4	30	WFG4
CDAS-SMPSO	4	30	WFG5
CDAS-SMPSO	4	30	WFG6
CDAS-SMPSO	4	30	WFG7
CDAS-SMPSO	4	30	WFG8
CDAS-SMPSO	4	30	WFG9
CDAS-SMPSO	6	30	WFG1
CDAS-SMPSO	6	30	WFG2
CDAS-SMPSO	6	30	WFG3
CDAS-SMPSO	6	30	WFG4
CDAS-SMPSO	6	30	WFG5
CDAS-SMPSO	6	30	WFG6
CDAS-SMPSO	6	30	WFG7
CDAS-SMPSO	6	30	WFG8

Continued on next page

Table A.1: Experiments Performed (continued)

Algorithm	**Objectives**	**Decision Variables**	**Function**
CDAS-SMPSO	6	30	WFG9
CDAS-SMPSO	8	30	WFG1
CDAS-SMPSO	8	30	WFG2
CDAS-SMPSO	8	30	WFG3
CDAS-SMPSO	8	30	WFG4
CDAS-SMPSO	8	30	WFG5
CDAS-SMPSO	8	30	WFG6
CDAS-SMPSO	8	30	WFG7
CDAS-SMPSO	8	30	WFG8
CDAS-SMPSO	8	30	WFG9
CDAS-SMPSO	10	30	WFG1
CDAS-SMPSO	10	30	WFG2
CDAS-SMPSO	10	30	WFG3
CDAS-SMPSO	10	30	WFG4
CDAS-SMPSO	10	30	WFG5
CDAS-SMPSO	10	30	WFG6
CDAS-SMPSO	10	30	WFG7

Continued on next page

Table A.1: Experiments Performed (continued)

Algorithm	**Objectives**	**Decision Variables**	**Function**
CDAS-SMPSO	10	30	WFG8
CDAS-SMPSO	10	30	WFG9
CDAS-SMPSO	10	100	WFG1
CDAS-SMPSO	10	100	WFG2
CDAS-SMPSO	10	100	WFG3
CDAS-SMPSO	10	100	WFG4
CDAS-SMPSO	10	100	WFG5
CDAS-SMPSO	10	100	WFG6
CDAS-SMPSO	10	100	WFG7
CDAS-SMPSO	10	100	WFG8
CDAS-SMPSO	10	100	WFG9
CDAS-SMPSO	10	500	WFG1
CDAS-SMPSO	10	500	WFG2
CDAS-SMPSO	10	500	WFG3
CDAS-SMPSO	10	500	WFG4
CDAS-SMPSO	10	500	WFG5
CDAS-SMPSO	10	500	WFG6

Continued on next page

Table A.1: Experiments Performed (continued)

Algorithm	**Objectives**	**Decision Variables**	**Function**
CDAS-SMPSO	10	500	WFG7
CDAS-SMPSO	10	500	WFG8
CDAS-SMPSO	10	500	WFG9
CDAS-SMPSO	10	1000	WFG1
CDAS-SMPSO	10	1000	WFG2
CDAS-SMPSO	10	1000	WFG3
CDAS-SMPSO	10	1000	WFG4
CDAS-SMPSO	10	1000	WFG5
CDAS-SMPSO	10	1000	WFG6
CDAS-SMPSO	10	1000	WFG7
CDAS-SMPSO	10	1000	WFG8
CDAS-SMPSO	10	1000	WFG9
SrEA/D	2	30	WFG1
SrEA/D	2	30	WFG2
SrEA/D	2	30	WFG3
SrEA/D	2	30	WFG4
SrEA/D	2	30	WFG5

Continued on next page

Table A.1: Experiments Performed (continued)

Algorithm	**Objectives**	**Decision Variables**	**Function**
SrEA/D	2	30	WFG6
SrEA/D	2	30	WFG7
SrEA/D	2	30	WFG8
SrEA/D	2	30	WFG9
SrEA/D	4	30	WFG1
SrEA/D	4	30	WFG2
SrEA/D	4	30	WFG3
SrEA/D	4	30	WFG4
SrEA/D	4	30	WFG5
SrEA/D	4	30	WFG6
SrEA/D	4	30	WFG7
SrEA/D	4	30	WFG8
SrEA/D	4	30	WFG9
SrEA/D	6	30	WFG1
SrEA/D	6	30	WFG2
SrEA/D	6	30	WFG3
SrEA/D	6	30	WFG4

Continued on next page

Table A.1: Experiments Performed (continued)

Algorithm	**Objectives**	**Decision Variables**	**Function**
SrEA/D	6	30	WFG5
SrEA/D	6	30	WFG6
SrEA/D	6	30	WFG7
SrEA/D	6	30	WFG8
SrEA/D	6	30	WFG9
SrEA/D	8	30	WFG1
SrEA/D	8	30	WFG2
SrEA/D	8	30	WFG3
SrEA/D	8	30	WFG4
SrEA/D	8	30	WFG5
SrEA/D	8	30	WFG6
SrEA/D	8	30	WFG7
SrEA/D	8	30	WFG8
SrEA/D	8	30	WFG9
SrEA/D	10	30	WFG1
SrEA/D	10	30	WFG2
SrEA/D	10	30	WFG3

Continued on next page

Table A.1: Experiments Performed (continued)

Algorithm	**Objectives**	**Decision Variables**	**Function**
SrEA/D	10	30	WFG4
SrEA/D	10	30	WFG5
SrEA/D	10	30	WFG6
SrEA/D	10	30	WFG7
SrEA/D	10	30	WFG8
SrEA/D	10	30	WFG9
SrEA/D	10	100	WFG1
SrEA/D	10	100	WFG2
SrEA/D	10	100	WFG3
SrEA/D	10	100	WFG4
SrEA/D	10	100	WFG5
SrEA/D	10	100	WFG6
SrEA/D	10	100	WFG7
SrEA/D	10	100	WFG8
SrEA/D	10	100	WFG9
SrEA/D	10	500	WFG1
SrEA/D	10	500	WFG2

Continued on next page

Table A.1: Experiments Performed (continued)

Algorithm	**Objectives**	**Decision Variables**	**Function**
SrEA/D	10	500	WFG3
SrEA/D	10	500	WFG4
SrEA/D	10	500	WFG5
SrEA/D	10	500	WFG6
SrEA/D	10	500	WFG7
SrEA/D	10	500	WFG8
SrEA/D	10	500	WFG9
SrEA/D	10	1000	WFG1
SrEA/D	10	1000	WFG2
SrEA/D	10	1000	WFG3
SrEA/D	10	1000	WFG4
SrEA/D	10	1000	WFG5
SrEA/D	10	1000	WFG6
SrEA/D	10	1000	WFG7
SrEA/D	10	1000	WFG8
SrEA/D	10	1000	WFG9
KnPSO	2	30	WFG1

Continued on next page

Table A.1: Experiments Performed (continued)

Algorithm	**Objectives**	**Decision Variables**	**Function**
KnPSO	2	30	WFG2
KnPSO	2	30	WFG3
KnPSO	2	30	WFG4
KnPSO	2	30	WFG5
KnPSO	2	30	WFG6
KnPSO	2	30	WFG7
KnPSO	2	30	WFG8
KnPSO	2	30	WFG9
KnPSO	4	30	WFG1
KnPSO	4	30	WFG2
KnPSO	4	30	WFG3
KnPSO	4	30	WFG4
KnPSO	4	30	WFG5
KnPSO	4	30	WFG6
KnPSO	4	30	WFG7
KnPSO	4	30	WFG8
KnPSO	4	30	WFG9

Continued on next page

Table A.1: Experiments Performed (continued)

Algorithm	**Objectives**	**Decision Variables**	**Function**
KnPSO	6	30	WFG1
KnPSO	6	30	WFG2
KnPSO	6	30	WFG3
KnPSO	6	30	WFG4
KnPSO	6	30	WFG5
KnPSO	6	30	WFG6
KnPSO	6	30	WFG7
KnPSO	6	30	WFG8
KnPSO	6	30	WFG9
KnPSO	8	30	WFG1
KnPSO	8	30	WFG2
KnPSO	8	30	WFG3
KnPSO	8	30	WFG4
KnPSO	8	30	WFG5
KnPSO	8	30	WFG6
KnPSO	8	30	WFG7
KnPSO	8	30	WFG8

Continued on next page

Table A.1: Experiments Performed (continued)

Algorithm	**Objectives**	**Decision Variables**	**Function**
KnPSO	8	30	WFG9
KnPSO	10	30	WFG1
KnPSO	10	30	WFG2
KnPSO	10	30	WFG3
KnPSO	10	30	WFG4
KnPSO	10	30	WFG5
KnPSO	10	30	WFG6
KnPSO	10	30	WFG7
KnPSO	10	30	WFG8
KnPSO	10	30	WFG9
KnPSO	10	100	WFG1
KnPSO	10	100	WFG2
KnPSO	10	100	WFG3
KnPSO	10	100	WFG4
KnPSO	10	100	WFG5
KnPSO	10	100	WFG6
KnPSO	10	100	WFG7

Continued on next page

Table A.1: Experiments Performed (continued)

Algorithm	**Objectives**	**Decision Variables**	**Function**
KnPSO	10	100	WFG8
KnPSO	10	100	WFG9
KnPSO	10	500	WFG1
KnPSO	10	500	WFG2
KnPSO	10	500	WFG3
KnPSO	10	500	WFG4
KnPSO	10	500	WFG5
KnPSO	10	500	WFG6
KnPSO	10	500	WFG7
KnPSO	10	500	WFG8
KnPSO	10	500	WFG9
KnPSO	10	1000	WFG1
KnPSO	10	1000	WFG2
KnPSO	10	1000	WFG3
KnPSO	10	1000	WFG4
KnPSO	10	1000	WFG5
KnPSO	10	1000	WFG6

Continued on next page

Table A.1: Experiments Performed (continued)

Algorithm	**Objectives**	**Decision Variables**	**Function**
KnPSO	10	1000	WFG7
KnPSO	10	1000	WFG8
KnPSO	10	1000	WFG9
KnEA	2	30	WFG1
KnEA	2	30	WFG2
KnEA	2	30	WFG3
KnEA	2	30	WFG4
KnEA	2	30	WFG5
KnEA	2	30	WFG6
KnEA	2	30	WFG7
KnEA	2	30	WFG8
KnEA	2	30	WFG9
KnEA	4	30	WFG1
KnEA	4	30	WFG2
KnEA	4	30	WFG3
KnEA	4	30	WFG4
KnEA	4	30	WFG5

Continued on next page

Table A.1: Experiments Performed (continued)

Algorithm	**Objectives**	**Decision Variables**	**Function**
KnEA	4	30	WFG6
KnEA	4	30	WFG7
KnEA	4	30	WFG8
KnEA	4	30	WFG9
KnEA	6	30	WFG1
KnEA	6	30	WFG2
KnEA	6	30	WFG3
KnEA	6	30	WFG4
KnEA	6	30	WFG5
KnEA	6	30	WFG6
KnEA	6	30	WFG7
KnEA	6	30	WFG8
KnEA	6	30	WFG9
KnEA	8	30	WFG1
KnEA	8	30	WFG2
KnEA	8	30	WFG3
KnEA	8	30	WFG4

Continued on next page

Table A.1: Experiments Performed (continued)

Algorithm	**Objectives**	**Decision Variables**	**Function**
KnEA	8	30	WFG5
KnEA	8	30	WFG6
KnEA	8	30	WFG7
KnEA	8	30	WFG8
KnEA	8	30	WFG9
KnEA	10	30	WFG1
KnEA	10	30	WFG2
KnEA	10	30	WFG3
KnEA	10	30	WFG4
KnEA	10	30	WFG5
KnEA	10	30	WFG6
KnEA	10	30	WFG7
KnEA	10	30	WFG8
KnEA	10	30	WFG9
KnEA	10	100	WFG1
KnEA	10	100	WFG2
KnEA	10	100	WFG3

Continued on next page

Table A.1: Experiments Performed (continued)

Algorithm	**Objectives**	**Decision Variables**	**Function**
KnEA	10	100	WFG4
KnEA	10	100	WFG5
KnEA	10	100	WFG6
KnEA	10	100	WFG7
KnEA	10	100	WFG8
KnEA	10	100	WFG9
KnEA	10	500	WFG1
KnEA	10	500	WFG2
KnEA	10	500	WFG3
KnEA	10	500	WFG4
KnEA	10	500	WFG5
KnEA	10	500	WFG6
KnEA	10	500	WFG7
KnEA	10	500	WFG8
KnEA	10	500	WFG9
KnEA	10	1000	WFG1
KnEA	10	1000	WFG2

Continued on next page

Table A.1: Experiments Performed (continued)

Algorithm	**Objectives**	**Decision Variables**	**Function**
KnEA	10	1000	WFG3
KnEA	10	1000	WFG4
KnEA	10	1000	WFG5
KnEA	10	1000	WFG6
KnEA	10	1000	WFG7
KnEA	10	1000	WFG8
KnEA	10	1000	WFG9
KnDE	2	30	WFG1
KnDE	2	30	WFG2
KnDE	2	30	WFG3
KnDE	2	30	WFG4
KnDE	2	30	WFG5
KnDE	2	30	WFG6
KnDE	2	30	WFG7
KnDE	2	30	WFG8
KnDE	2	30	WFG9
KnDE	4	30	WFG1

Continued on next page

Table A.1: Experiments Performed (continued)

Algorithm	**Objectives**	**Decision Variables**	**Function**
KnDE	4	30	WFG2
KnDE	4	30	WFG3
KnDE	4	30	WFG4
KnDE	4	30	WFG5
KnDE	4	30	WFG6
KnDE	4	30	WFG7
KnDE	4	30	WFG8
KnDE	4	30	WFG9
KnDE	6	30	WFG1
KnDE	6	30	WFG2
KnDE	6	30	WFG3
KnDE	6	30	WFG4
KnDE	6	30	WFG5
KnDE	6	30	WFG6
KnDE	6	30	WFG7
KnDE	6	30	WFG8
KnDE	6	30	WFG9

Continued on next page

Table A.1: Experiments Performed (continued)

Algorithm	**Objectives**	**Decision Variables**	**Function**
KnDE	8	30	WFG1
KnDE	8	30	WFG2
KnDE	8	30	WFG3
KnDE	8	30	WFG4
KnDE	8	30	WFG5
KnDE	8	30	WFG6
KnDE	8	30	WFG7
KnDE	8	30	WFG8
KnDE	8	30	WFG9
KnDE	10	30	WFG1
KnDE	10	30	WFG2
KnDE	10	30	WFG3
KnDE	10	30	WFG4
KnDE	10	30	WFG5
KnDE	10	30	WFG6
KnDE	10	30	WFG7
KnDE	10	30	WFG8

Continued on next page

Table A.1: Experiments Performed (continued)

Algorithm	**Objectives**	**Decision Variables**	**Function**
KnDE	10	30	WFG9
KnDE	10	100	WFG1
KnDE	10	100	WFG2
KnDE	10	100	WFG3
KnDE	10	100	WFG4
KnDE	10	100	WFG5
KnDE	10	100	WFG6
KnDE	10	100	WFG7
KnDE	10	100	WFG8
KnDE	10	100	WFG9
KnDE	10	500	WFG1
KnDE	10	500	WFG2
KnDE	10	500	WFG3
KnDE	10	500	WFG4
KnDE	10	500	WFG5
KnDE	10	500	WFG6
KnDE	10	500	WFG7

Continued on next page

Table A.1: Experiments Performed (continued)

Algorithm	**Objectives**	**Decision Variables**	**Function**
KnDE	10	500	WFG8
KnDE	10	500	WFG9
KnDE	10	1000	WFG1
KnDE	10	1000	WFG2
KnDE	10	1000	WFG3
KnDE	10	1000	WFG4
KnDE	10	1000	WFG5
KnDE	10	1000	WFG6
KnDE	10	1000	WFG7
KnDE	10	1000	WFG8
KnDE	10	1000	WFG9
NSGA-III	2	30	WFG1
NSGA-III	2	30	WFG2
NSGA-III	2	30	WFG3
NSGA-III	2	30	WFG4
NSGA-III	2	30	WFG5
NSGA-III	2	30	WFG6

Continued on next page

Table A.1: Experiments Performed (continued)

Algorithm	**Objectives**	**Decision Variables**	**Function**
NSGA-III	2	30	WFG7
NSGA-III	2	30	WFG8
NSGA-III	2	30	WFG9
NSGA-III	4	30	WFG1
NSGA-III	4	30	WFG2
NSGA-III	4	30	WFG3
NSGA-III	4	30	WFG4
NSGA-III	4	30	WFG5
NSGA-III	4	30	WFG6
NSGA-III	4	30	WFG7
NSGA-III	4	30	WFG8
NSGA-III	4	30	WFG9
NSGA-III	6	30	WFG1
NSGA-III	6	30	WFG2
NSGA-III	6	30	WFG3
NSGA-III	6	30	WFG4
NSGA-III	6	30	WFG5

Continued on next page

Table A.1: Experiments Performed (continued)

Algorithm	**Objectives**	**Decision Variables**	**Function**
NSGA-III	6	30	WFG6
NSGA-III	6	30	WFG7
NSGA-III	6	30	WFG8
NSGA-III	6	30	WFG9
NSGA-III	8	30	WFG1
NSGA-III	8	30	WFG2
NSGA-III	8	30	WFG3
NSGA-III	8	30	WFG4
NSGA-III	8	30	WFG5
NSGA-III	8	30	WFG6
NSGA-III	8	30	WFG7
NSGA-III	8	30	WFG8
NSGA-III	8	30	WFG9
NSGA-III	10	30	WFG1
NSGA-III	10	30	WFG2
NSGA-III	10	30	WFG3
NSGA-III	10	30	WFG4

Continued on next page

Table A.1: Experiments Performed (continued)

Algorithm	**Objectives**	**Decision Variables**	**Function**
NSGA-III	10	30	WFG5
NSGA-III	10	30	WFG6
NSGA-III	10	30	WFG7
NSGA-III	10	30	WFG8
NSGA-III	10	30	WFG9
NSGA-III	10	100	WFG1
NSGA-III	10	100	WFG2
NSGA-III	10	100	WFG3
NSGA-III	10	100	WFG4
NSGA-III	10	100	WFG5
NSGA-III	10	100	WFG6
NSGA-III	10	100	WFG7
NSGA-III	10	100	WFG8
NSGA-III	10	100	WFG9
NSGA-III	10	500	WFG1
NSGA-III	10	500	WFG2
NSGA-III	10	500	WFG3

Continued on next page

Table A.1: Experiments Performed (continued)

Algorithm	**Objectives**	**Decision Variables**	**Function**
NSGA-III	10	500	WFG4
NSGA-III	10	500	WFG5
NSGA-III	10	500	WFG6
NSGA-III	10	500	WFG7
NSGA-III	10	500	WFG8
NSGA-III	10	500	WFG9
NSGA-III	10	1000	WFG1
NSGA-III	10	1000	WFG2
NSGA-III	10	1000	WFG3
NSGA-III	10	1000	WFG4
NSGA-III	10	1000	WFG5
NSGA-III	10	1000	WFG6
NSGA-III	10	1000	WFG7
NSGA-III	10	1000	WFG8
NSGA-III	10	1000	WFG9
MOEA/D	2	30	WFG1
MOEA/D	2	30	WFG2

Continued on next page

Table A.1: Experiments Performed (continued)

Algorithm	**Objectives**	**Decision Variables**	**Function**
MOEA/D	2	30	WFG3
MOEA/D	2	30	WFG4
MOEA/D	2	30	WFG5
MOEA/D	2	30	WFG6
MOEA/D	2	30	WFG7
MOEA/D	2	30	WFG8
MOEA/D	2	30	WFG9
MOEA/D	4	30	WFG1
MOEA/D	4	30	WFG2
MOEA/D	4	30	WFG3
MOEA/D	4	30	WFG4
MOEA/D	4	30	WFG5
MOEA/D	4	30	WFG6
MOEA/D	4	30	WFG7
MOEA/D	4	30	WFG8
MOEA/D	4	30	WFG9
MOEA/D	6	30	WFG1

Continued on next page

Table A.1: Experiments Performed (continued)

Algorithm	**Objectives**	**Decision Variables**	**Function**
MOEA/D	6	30	WFG2
MOEA/D	6	30	WFG3
MOEA/D	6	30	WFG4
MOEA/D	6	30	WFG5
MOEA/D	6	30	WFG6
MOEA/D	6	30	WFG7
MOEA/D	6	30	WFG8
MOEA/D	6	30	WFG9
MOEA/D	8	30	WFG1
MOEA/D	8	30	WFG2
MOEA/D	8	30	WFG3
MOEA/D	8	30	WFG4
MOEA/D	8	30	WFG5
MOEA/D	8	30	WFG6
MOEA/D	8	30	WFG7
MOEA/D	8	30	WFG8
MOEA/D	8	30	WFG9

Continued on next page

Table A.1: Experiments Performed (continued)

Algorithm	**Objectives**	**Decision Variables**	**Function**
MOEA/D	10	30	WFG1
MOEA/D	10	30	WFG2
MOEA/D	10	30	WFG3
MOEA/D	10	30	WFG4
MOEA/D	10	30	WFG5
MOEA/D	10	30	WFG6
MOEA/D	10	30	WFG7
MOEA/D	10	30	WFG8
MOEA/D	10	30	WFG9
MOEA/D	10	100	WFG1
MOEA/D	10	100	WFG2
MOEA/D	10	100	WFG3
MOEA/D	10	100	WFG4
MOEA/D	10	100	WFG5
MOEA/D	10	100	WFG6
MOEA/D	10	100	WFG7
MOEA/D	10	100	WFG8

Continued on next page

Table A.1: Experiments Performed (continued)

Algorithm	**Objectives**	**Decision Variables**	**Function**
MOEA/D	10	100	WFG9
MOEA/D	10	500	WFG1
MOEA/D	10	500	WFG2
MOEA/D	10	500	WFG3
MOEA/D	10	500	WFG4
MOEA/D	10	500	WFG5
MOEA/D	10	500	WFG6
MOEA/D	10	500	WFG7
MOEA/D	10	500	WFG8
MOEA/D	10	500	WFG9
MOEA/D	10	1000	WFG1
MOEA/D	10	1000	WFG2
MOEA/D	10	1000	WFG3
MOEA/D	10	1000	WFG4
MOEA/D	10	1000	WFG5
MOEA/D	10	1000	WFG6
MOEA/D	10	1000	WFG7

Continued on next page

Table A.1: Experiments Performed (continued)

Algorithm	**Objectives**	**Decision Variables**	**Function**
MOEA/D	10	1000	WFG8
MOEA/D	10	1000	WFG9